Titre: Title:	Méthodes itératives pour la résolution par éléments finis d'écoulements à surfaces libres
Auteur: Author:	Choyebe Alain Fidahoussen
Date:	2008
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Fidahoussen, C. A. (2008). Méthodes itératives pour la résolution par éléments finis d'écoulements à surfaces libres [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8213/

UNIVERSITÉ DE MONTRÉAL

MÉTHODES ITÉRATIVES POUR LA RÉSOLUTION PAR ÉLÉMENTS FINIS D'ÉCOULEMENTS À SURFACES LIBRES

CHOYEBE ALAIN FIDAHOUSSEN
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
DÉCEMBRE 2008

Your file Votre référence
ISBN: 978-0-494-48916-1
Our file Notre référence
ISBN: 978-0-494-48916-1

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

MÉTHODES ITÉRATIVES POUR LA RÉSOLUTION PAR ÉLÉMENTS FINIS D'ÉCOULEMENTS À SURFACES LIBRES

présenté par : FIDAHOUSSEN Choyebe Alain

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. SAUCIER Antoine, Ph.D, président

M. DUFOUR Steven, Ph.D, membre et directeur de recherche

M. ORBAN Dominique, Doct. Sc., membre et codirecteur de recherche

M. BERTRAND François, Ph.D, membre

À Thomas W.

REMERCIEMENTS

Ce mémoire n'aurait pas connu le jour sans le soutien de mes directeurs, et c'est donc à eux que mes premiers remerciements reviendront. Merci à Steven Dufour et à Dominique Orban pour m'avoir aidé, financé et guidé tout le long de ce projet de recherche. Merci également à Charles, Yvan, Donatien et Antoine pour leur bonne humeur qui a agrémenté ces longues heures passées dans le « bocal » étudiant, ainsi que pour leurs conseils avisés qui m'ont permis de passer outre un grand nombre de difficultés. J'adresse pour finir une pensée chaleureuse envers ma famille proche, trop éloignée, et mes amis, ceux qui sont restés malgré tout et ceux qui sont partis malgré moi.

RÉSUMÉ

Nous nous intéressons dans ce mémoire à la résolution des systèmes linéaires issus de la discrétisation éléments-finis des équations modélisant les écoulements de fluides incompressibles et non miscibles. L'interface entre les fluides est une variable du problème. Nous appliquons pour cela un schéma de Newton-Krylov à critère d'arrêt adaptatif sur la formulation mixte. Nous introduisons une nouvelle classe de méthodes itératives particulièrement adaptées à la structure des systèmes linéaires rencontrés à chaque itération de Newton. Ces méthodes sont inspirées de l'algorithme du gradient conjugué projeté, fréquemment utilisé dans les problèmes quadratiques, et travaillent implicitement dans le noyau de l'opérateur divergence. Elles requièrent uniquement des produits matrice-vecteur avec la matrice de convection-diffusion et ne demandent qu'une seule factorisation symétrique indéfinie de la matrice de projection. Ces deux caractéristiques ont pour conséquence des gains importants en termes de coût de calcul et d'espace mémoire. Nous validons d'abord notre approche sur des problèmes test mettant en jeu un ou deux fluides. Dans un second temps, nous présentons trois problèmes pouvant être résolus par notre nouvelle classe de méthodes itératives : les écoulements non isothermes, les fluides visco-élastiques et la méthode des domaines fictifs pour modéliser les écoulements avec frontières mobiles. Cette approche possède l'avantage de construire une discrétisation des frontières indépendante du maillage éléments-finis. Ce dernier n'est donc généré qu'une seule fois.

ABSTRACT

In this thesis, we consider the solution of linear systems arising from the finite-element discretization of incompressible and non miscible flows. The location of the interface between the fluids is a variable of the problem. Our approach is to use a Newton-Krylov scheme with adaptive stopping tolerance on the mixed formulation. We introduce a new class of iterative methods specifically tailored for the structure of the linear system encountered at each Newton iteration. Those methods draw from the projected conjugate gradient method, often used in quadratic programming, and implicitly work in the nullspace of the divergence operator. They only require matrix-vector products with the convection-diffusion matrix and a one-time symmetric indefinite factorization of a projection matrix. This results in important gains in terms of computational cost and memory requirements. We first validate our approach on test problems with one and two fluids. In a second stage, we present three applications that can be solved with our new class of iterative methods : non isothermal fluid flows, viscoelastic fluids and flows with moving boundaries using the fictitious domains method. This last method has the advantage that the discretization of the boundaries is independent of the finite-element mesh. The mesh is thus only generated once.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

INTRODUCTION

Le contexte de ce projet de recherche est la simulation numérique d'écoulements à surfaces libres de fluides incompressibles et non miscibles. À la base de la modélisation de ces écoulements, les équations de Navier-Stokes traduisent mathématiquement la conservation de la quantité de mouvement et la conservation de la masse. Ces équations, exceptés pour des cas simples menant à des simplifications, n'ont pas de solutions analytiques connues à ce jour. Les équations de Navier-Stokes font d'ailleurs parties d'une liste de sept problèmes, dit « du millénaire », établis par le *Clay Mathematics Institute* et dont la résolution sera récompensée par une forte somme. La mise en place de stratégies de résolution numériques efficaces est donc primordiale, d'autant plus que des équations supplémentaires peuvent venir se rajouter pour certains types d'écoulements. Par exemple, lors d'un écoulement multiphase, une équation de transport doit être ajoutée pour modéliser la dynamique de l'interface entre les fluides. Lorsqu'un écoulement n'est pas isotherme, l'équation de conservation de l'énergie doit être prise en compte. Pour des fluides visco-élastiques, on rajoute une loi dite *de comportement.*

Plusieurs stratégies numériques sont envisageables pour résoudre des équations aux dérivées partielles avec conditions aux limites. Une des plus populaires est la méthode des éléments finis qui consiste à approcher la solution sur un espace de dimension finie à l'aide d'un maillage du domaine d'écoulement. L'obtention du résultat passe par la résolution d'un ou de plusieurs systèmes matriciels et des algorithmes adaptés doivent dès lors être envisagés. Par « algorithmes adaptés » nous entendons des méthodes rapides qui nécessitent peu d'espace mémoire. Ces deux critères prennent leur importance notamment lorsque les fluides s'écoulent dans des géométries complexes qui peuvent mener à des maillages fins et/ou raffinés en certains endroits, ce qui mène à des matrices de grande taille. Deux familles de méthodes

peuvent être utilisées : les méthodes directes et les méthodes itératives, chacune possédant ses avantages et ses inconvénients en terme de précision des résultats, de stockage mémoire et de rapidité.

Le principal objectif de ce projet de recherche est d'implémenter et tester une nouvelle classe de méthodes, à mi-chemin entre les méthodes directes et itératives, permettant de résoudre efficacement les équations de Navier-Stokes dans un contexte de problèmes multi-physiques. De nombreux travaux concernant les méthodes itératives ont été effectués ces dernières décennies, menant notamment aux algorithmes de *gradient conjugué* (GC) (Hestenes 1952), *biconjugate gradient stabilized* (BiCGSTAB) (Van Der Vorst 1992), *generalized minimal residual* (GMRES) (Saad and Schultz 1984) et *transpose-free quasi-minimal residual* (TFQMR) (Freund and Nachtigal 1994) pour n'en citer que quelques-uns. De cet objectif principal, nous avons défini les objectifs spécifiques suivants :

- appliquer la méthode du gradient conjugué projeté aux systèmes linéaires issus de la discrétisation éléments-finis des équations de Stokes ;
- implémenter et tester d'autres méthodes itératives projetées pour résoudre les équations de Navier-Stokes ;
- étudier d'autres problèmes d'écoulements pouvant être résolus à l'aide des méthodes projetées.

Dans un premier temps, nous avons considéré les équations de Stokes qui sont un cas particulier des équations de Navier-Stokes, correspondant aux écoulements à faible nombre de Reynolds. Ce choix est motivé par le fait que la discrétisation du terme de diffusion de l'équation de conservation de la quantité de mouvement aboutit à une matrice symétrique définie positive. Les équations de Stokes discrétisées peuvent alors être vues comme des problèmes d'optimisation quadratique avec contraintes d'égalité. L'algorithme du gradient conjugué projeté, performant pour résoudre ce type de problèmes, peut alors être utilisé (Gould et al. 1998).

Dès lors, nous nous sommes interrogés dans un second temps sur une possible généralisation aux autres systèmes augmentés découlant des équations de Navier-Stokes. En effet, comparativement aux équations de Stokes, un terme de convection supplémentaire est à prendre en compte. Ce dernier a pour incidence que la matrice obtenue par la discrétisation éléments-finis n'est plus symétrique. Il n'y a alors plus d'interprétation en terme de problème d'optimisation. Pour remplir ce deuxième objectif, nous nous sommes demandés si, à l'image du gradient conjugué, nous pouvions projeter un autre algorithme. La performance d'une méthode itérative étant reliée au problème considéré, nous avons testé les différents algorithmes utilisés dans la littérature et sélectionné le plus efficace afin d'implémenter sa version projetée. La difficulté de la résolution d'un problème d'écoulement se mesure essentiellement par le conditionnement du bloc matriciel correspondant à la discrétisation des termes de convection et de diffusion de l'équation de conservation de la quantité de mouvement. De plus, les méthodes projetées requièrent des produits matrice-vecteur avec ce bloc matriciel. C'est pour ces deux raisons que nos tests ont été effectués sur ce bloc et non sur la matrice au complet.

Enfin, dans un dernier temps, nous nous sommes posés la question de savoir si d'autres problèmes d'écoulements fréquemment retrouvés dans la littérature pouvaient être résolus par notre nouvelle classe de méthodes itératives. En d'autres termes, nous avons formé, pour chacun de ces problèmes, les systèmes linéaires découlant de la discrétisation par la méthode des éléments finis des équations de modélisation, et nous avons vérifié que les matrices gardaient leur structure augmentée.

Ce mémoire est structuré de la façon suivante. Le premier chapitre nous permet de présenter les équations mathématiques modélisant les écoulements multifluides à surface libre, ainsi que leur discrétisation par la méthode des éléments finis. Cette discrétisation aboutit à un système linéaire qui peut être résolu par des méthodes

directes ou itératives. Ces dernières font l'objet du deuxième chapitre. Après y avoir exposé les principaux algorithmes fréquemment trouvés dans la littérature, nous présentons deux nouveaux algorithmes, dits projetés, adaptés respectivement aux équations de Stokes et de Navier-Stokes. Les résultats numériques obtenus par l'utilisation de ces algorithmes sur trois problèmes sont donnés dans le troisième chapitre. Des comparaisons de performance avec les algorithmes directs, en termes de temps de calcul et de stockage mémoire, sont effectuées. Enfin, dans le quatrième chapitre, nous présentons trois exemples d'application des algorithmes projetés : les écoulements non isothermes, les écoulements de fluides visco-élastiques et les écoulements avec frontières mobiles avec la méthode des domaines fictifs.

CHAPITRE 1

MODÉLISATION DES ÉCOULEMENTS MULTIFLUIDES

Ce premier chapitre va nous permettre de présenter les équations mathématiques modélisant des écoulements ayant les caractéristiques suivantes :

- **multifluides** : mettant en jeu plusieurs fluides non miscibles. La frontière entre les fluides est libre et fait partie des inconnues du problème ;
- **incompressibles** : à volume quasiment constant sous l'action des pressions externes.

Nous utilisons pour modéliser ces écoulements les équations de Stokes ou de Navier-Stokes. À ces équations, qui symbolisent la conservation de la quantité de mouvement, il faut ajouter la conservation de la masse et la modélisation de la dynamique de la surface libre. Ce système est complété par des conditions aux limites et initiales appropriées.

Dans un premier temps, ce chapitre va nous permettre de présenter la modélisation du problème continu décrit plus haut. Dans un second temps, nous nous pencherons sur les différentes étapes qui aboutissent au problème discret, *i.e.* la formulation variationnelle, la linéarisation des équations et leurs discrétisations en espace et en temps. Enfin, dans un dernier temps, nous présenterons deux stratégies de résolution du système provenant de la discrétisation des équations aux dérivées partielles : la méthode mixte et la méthode d'Uzawa.

1.1 Modélisation des écoulements incompressibles : problème continu

1.1.1 Équations de modélisation

La modélisation de l'écoulement d'un fluide incompressible s'effectue par l'intermédiaire de deux équations qui expriment mathématiquement les deux lois physiques que sont la conservation de la masse et la conservation de la quantité de mouvement. La première est l'équation de continuité et traduit la loi physique suivante :

$$\begin{pmatrix} \textit{flux entrant} \\ \textit{dans } \mathcal{V} \end{pmatrix} - \begin{pmatrix} \textit{flux sortant} \\ \textit{de } \mathcal{V} \end{pmatrix} = \begin{pmatrix} \textit{variation de masse dans } \mathcal{V} \\ \textit{par unité de temps} \end{pmatrix},$$

où $\mathcal{V}$ est un volume infinitésimal. Cette loi physique se traduit mathématiquement par (Chorin and Marsden 1992) :

$$\frac{\partial \rho_i}{\partial t} + \boldsymbol{\nabla} \cdot (\rho_i \boldsymbol{u}_i) = 0,$$

où $\boldsymbol{u}_i = (u_i \ v_i)^T$ est le vecteur vitesse lié à l'écoulement du fluide i et ρ_i est sa densité.

La deuxième loi physique est le second principe fondamental de la dynamique, i.e. :

$$\begin{pmatrix} \textit{somme des forces} \\ \textit{exercées sur } \mathcal{V} \end{pmatrix} = \begin{pmatrix} \textit{variation de la quantité de mouvement} \\ \textit{dans } \mathcal{V} \textit{ par unité de temps} \end{pmatrix},$$

ce qui se traduit mathématiquement par (Chorin and Marsden 1992) :

$$\rho_i \left(\frac{\partial \boldsymbol{u}_i}{\partial t} + (\boldsymbol{u}_i \cdot \boldsymbol{\nabla})\boldsymbol{u}_i\right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}_i = \rho \boldsymbol{f}_i,$$

où $\rho_i \boldsymbol{f}_i$ est un vecteur de forces volumiques supposé connu et $\boldsymbol{\sigma}_i$ est le tenseur des contraintes internes. Ce dernier, plus communément appelé *tenseur de Cauchy*, est défini comme suit :

$$\boldsymbol{\sigma}_i = -p_i \boldsymbol{I} + \boldsymbol{\tau}_i, \tag{1.1}$$

où p_i est la pression à l'intérieur du fluide i et $\boldsymbol{I}$ est le tenseur identité. Le *tenseur des extra-contraintes* $\boldsymbol{\tau}_i$ est relié au champ de vitesse $\boldsymbol{u}_i$ par la relation

$$\boldsymbol{\tau}_i = 2\mu_i \dot{\boldsymbol{\gamma}}_i(\boldsymbol{u}_i),$$

où μ_i est la viscosité constante du fluide i. Le tenseur $\dot{\boldsymbol{\gamma}}_i$ est le *tenseur vitesse-de-déformation* et est défini comme suit :

$$\dot{\boldsymbol{\gamma}}_i(\boldsymbol{u}_i) = \frac{1}{2}(\boldsymbol{\nabla}\boldsymbol{u}_i + (\boldsymbol{\nabla}\boldsymbol{u}_i)^T) = \begin{pmatrix} \dfrac{\partial u_i}{\partial x} & \dfrac{1}{2}\left(\dfrac{\partial u_i}{\partial y} + \dfrac{\partial v_i}{\partial x}\right) \\ \dfrac{1}{2}\left(\dfrac{\partial u_i}{\partial y} + \dfrac{\partial v_i}{\partial x}\right) & \dfrac{\partial v_i}{\partial y} \end{pmatrix}.$$

Notations :

1. L'opérateur différentiel $\boldsymbol{\nabla}$ est appliqué à l'aide de deux produits différents. Celui noté par un point est le produit scalaire ordinaire, de sorte que la quantité suivante est une divergence :

$$\boldsymbol{\nabla} \cdot \boldsymbol{u}_i = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} \cdot \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \frac{\partial u_i}{\partial x} + \frac{\partial v_i}{\partial y}.$$

   Par contre, le produit représenté sans le point est le produit tensoriel, ou produit direct, et l'expression suivante est alors un tenseur :

$$\boldsymbol{\nabla}\boldsymbol{u}_i = \boldsymbol{\nabla} \otimes \boldsymbol{u}_i = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} \begin{pmatrix} u_i & v_i \end{pmatrix} = \begin{pmatrix} \dfrac{\partial u_i}{\partial x} & \dfrac{\partial v_i}{\partial x} \\ \dfrac{\partial u_i}{\partial y} & \dfrac{\partial v_i}{\partial y} \end{pmatrix};$$

2. L'équation de conservation de la quantité de mouvement fait intervenir le produit $\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}$. Appliqué à un tenseur, l'opérateur $\boldsymbol{\nabla}$ revient à appliquer une

divergence sur chacune des colonnes de la matrice. Le résultat est donc un vecteur :

$$\boldsymbol{\nabla} \cdot \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial a_{11}}{\partial x} + \dfrac{\partial a_{21}}{\partial y} \\ \dfrac{\partial a_{12}}{\partial x} + \dfrac{\partial a_{22}}{\partial y} \end{pmatrix} ;$$

3. L'opérateur « : » est la double contraction et est défini comme suit :

$$\boldsymbol{\sigma} : \boldsymbol{\tau} = \sum_{k=1}^{2} \sum_{l=1}^{2} \boldsymbol{\sigma}_{kl} \boldsymbol{\tau}_{lk}.$$

Les écoulements qui nous intéressent sont incompressibles (ρ_i constante), de sorte que les équations de conservation s'écrivent dans notre cas :

- Conservation de la *masse* :

$$\underbrace{\boldsymbol{\nabla} \cdot \boldsymbol{u}_i}_{\textit{divergence}} = 0 ; \tag{1.2}$$

- Conservation de la *quantité de mouvement* :

$$\underbrace{\rho_i \frac{\partial \boldsymbol{u}_i}{\partial t}}_{\substack{\textit{terme} \\ \textit{transitoire}}} + \underbrace{\rho_i (\boldsymbol{u}_i \cdot \boldsymbol{\nabla}) \boldsymbol{u}_i}_{\textit{convection}} - \underbrace{\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}_i}_{\substack{\textit{diffusion} \\ \textit{visqueuse}}} = \underbrace{\rho_i \boldsymbol{f}_i}_{\substack{\textit{force} \\ \textit{volumique}}} . \tag{1.3}$$

Remarque : les équations de Stokes régissent les écoulements dans lesquels les effets visqueux prédominent sur les effets inertiels. Ces équations sont donc une version simplifiée des équations de Navier-Stokes, dans lequelles il suffit d'enlever le terme de convection $(\boldsymbol{u}_i \cdot \boldsymbol{\nabla}) \boldsymbol{u}_i$ de l'équation (1.3).

Il est nécessaire de compléter les équations (1.2) et (1.3) par des conditions aux limites et initiales adéquates. Les conditions aux limites peuvent être dites de Dirichlet ou de Neumann. Les premières portent sur les composantes de la vitesse $\boldsymbol{u}_i$,

alors que les secondes portent sur les composantes de la contrainte normale $\boldsymbol{\sigma}_i \cdot \boldsymbol{n}$ où $\boldsymbol{n}$ est la normale extérieure à la frontière $\Gamma = \partial\Omega$ du domaine de calcul Ω. Si l'on note Γ_{D_u} et Γ_{N_u} respectivement les domaines de définition des conditions de Dirichlet et de Neumann, les conditions aux limites et initiales s'écrivent alors :

$$\begin{cases} \boldsymbol{u}_i & = & \boldsymbol{u}_{D_u} & \text{sur } \Gamma_{D_u} \ ; \\ \boldsymbol{\sigma}_i \cdot \boldsymbol{n} & = & \boldsymbol{t}_{N_u} & \text{sur } \Gamma_{N_u}, \end{cases} \tag{1.4}$$

et

$$\boldsymbol{u}_i(\boldsymbol{x}, t = 0) = \boldsymbol{u}_i^0. \tag{1.5}$$

Les domaines Γ_{D_u} et Γ_{N_u} forment une partition de la frontière Γ et vérifient donc :

$$\begin{cases} \Gamma_{D_u} \cup \Gamma_{N_u} & = & \partial\Omega \ ; \\ \Gamma_{D_u} \cap \Gamma_{N_u} & = & \emptyset. \end{cases} \tag{1.6}$$

Remarque : l'intersection vide entre les domaines Γ_{D_u} et Γ_{N_u} exprime le fait qu'on ne peut pas imposer à la fois une condition de Dirichlet et une condition de Neumann sur une même partie de la frontière $\Gamma = \partial\Omega$. La troisième condition (1.5) est la vitesse initiale du fluide au temps $t = 0$.

1.1.2 Écoulements multifluides et conditions aux interfaces

Une surface libre est une région délimitant deux phases, solide, liquide ou gazeuse, qui peuvent éventuellement être de même nature. Nous rappelons ici qu'est faite l'hypothèse d'immiscibilité. Nous ne nous intéresserons dans ce mémoire qu'aux interfaces liquide-liquide et liquide-gaz.

Dans un problème multifluide, chaque phase se voit affectée d'un problème différent. Pour que le problème résultant soit bien posé, il est donc nécessaire que des conditions aux limites appropriées soient imposées aux interfaces. Considérons la figure 1.1. L'interface $\mathcal{S}$ représente la surface libre qui sépare les fluides 1 et 2. Chacun de ces deux fluides s'étend respectivement dans les domaines Ω_1 et Ω_2. Si l'on note $\boldsymbol{t}_S$ et $\boldsymbol{n}_S$

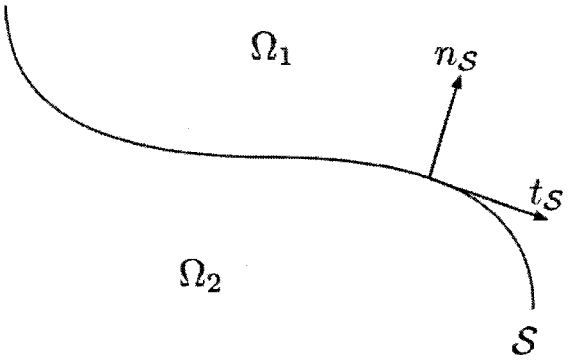


Figure 1.1 – Interface entre deux fluides.

respectivement le vecteur tangent et le vecteur normal à la surface libre, les conditions à l'interface (Batchelor 1967) sont alors données par la *continuité de la vitesse normale* :

$$\boldsymbol{u}_1 \cdot \boldsymbol{n}_S = \boldsymbol{u}_2 \cdot \boldsymbol{n}_S = \boldsymbol{u}_S \cdot \boldsymbol{n}_S, \tag{1.7}$$

où $\boldsymbol{u}_1$ et $\boldsymbol{u}_2$ sont les vitesses des fluides 1 et 2 alors que $\boldsymbol{u}_S$ est la vitesse de l'interface elle-même, la *continuité de la vitesse tangentielle* :

$$\boldsymbol{u}_1 \cdot \boldsymbol{t}_S = \boldsymbol{u}_2 \cdot \boldsymbol{t}_S, \tag{1.8}$$

et l'*équilibre des forces à l'interface* :

$$(\boldsymbol{\sigma}_2 - \boldsymbol{\sigma}_1) \cdot \boldsymbol{n}_S = \alpha\kappa\boldsymbol{n}_S, \tag{1.9}$$

où α est le coefficient de tension superficielle des fluides 1 et 2 et κ est la courbure locale de la surface libre. Nous ne considérons dans ce mémoire que des écoulements sans tension superficielle ($\alpha = 0$). Nous pouvons supposer, sans perte de généralité, que $\boldsymbol{n}_S$ pointe dans la direction du fluide 1, tel qu'illustré à la figure 1.1 et que le rayon de courbure est positif si le centre de courbure se trouve dans le fluide 1.

1.1.3 Équation de transport de la pseudo-concentration

Les fluides considérés étant non miscibles, la condition de continuité de la vitesse normale à l'interface (1.7) peut être interprétée comme une condition de non

pénétration de la surface libre. Dans le cas d'un écoulement stationnaire, cette condition devient

$$\boldsymbol{u}_{\mathcal{S}} \cdot \boldsymbol{n}_{\mathcal{S}} = 0. \tag{1.10}$$

Cette condition frontière n'est définie qu'à l'interface et n'est par conséquent pas très pratique à utiliser (la position de l'interface fait partie des inconnues du problème). Il serait plus facile de tirer parti d'une condition définie sur tout le domaine Ω. Définissons la fonction suivante :

$$F(\boldsymbol{x}) = \begin{cases} 1 & \text{si} \quad \boldsymbol{x} \in \Omega_1 \, ; \\ 0 & \text{si} \quad \boldsymbol{x} \in \Omega_2. \end{cases}$$

Cette fonction est connue sous le nom de *pseudo-concentration* des fluides 1 et 2 dans le domaine Ω (Thompson 1986). La position de l'interface $\mathcal{S}$ est donnée par l'ensemble des points $\boldsymbol{x}$ pour lesquels F prend pour valeur $\frac{1}{2}$.

À l'aide de la pseudo-concentration, nous pouvons maintenant définir une expression équivalente, en régime stationnaire, à la condition d'immisciblité (1.10), sous la forme d'une équation de transport (Agassant et al. 1994) :

$$\boldsymbol{u}_i \cdot \boldsymbol{\nabla} F = 0, \quad \forall \, \boldsymbol{x} \in \Omega, \quad \text{pour } i = 1, 2, \tag{1.11}$$

dont l'avantage est d'être définie sur tout le domaine Ω. En régime transitoire, où $F = F(\boldsymbol{x}, t)$, on a :

$$\frac{dF}{dt} = \frac{\partial F}{\partial x}\frac{dx}{dt} + \frac{\partial F}{\partial y}\frac{dy}{dt} + \frac{\partial F}{\partial t}\frac{dt}{dt} = \frac{\partial F}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} F.$$

Soit $\mathcal{V}$ un volume infinitésimal. Supposons que

$$\frac{d}{dt}\int_{\mathcal{V}} F dV = 0.$$

Or, d'après le théorème de Reynolds (Chorin and Marsden 1992) :

$$\frac{d}{dt}\int_{\mathcal{V}} F dV = \int_{\mathcal{V}} \left(\frac{\partial F}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} F\right) dV.$$

Donc :

$$\int_{\mathcal{V}} \left(\frac{\partial F}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} F\right) dV = 0. \tag{1.12}$$

La relation (1.12) étant vraie quelque soit le volume $\mathcal{V}$ choisi, on a alors :

$$\frac{\partial F}{\partial t} + \boldsymbol{u}_i \cdot \boldsymbol{\nabla} F = 0, \quad \forall\, \boldsymbol{x} \in \Omega, \quad \text{pour } i = 1, 2, \tag{1.13}$$

qui est l'équation de transport en régime transitoire. Avec les conditions aux limites et initiale appropriées appliquées à F, le saut de cette fonction identifiera la position de la surface libre dans Ω. Puisque l'équation de transport (1.13) est hyperbolique, les conditions aux limites ne doivent être appliquées qu'en amont de l'écoulement :

$$\Gamma^{-}(\Omega) = \{\boldsymbol{x} \in \partial\Omega \mid \boldsymbol{u}_i \cdot \boldsymbol{n} < 0\}, \tag{1.14}$$

et où $\boldsymbol{n}$ est la normale extérieure au domaine Ω. Nous avons donc la condition

$$F = F_{\Gamma^-} \quad \text{sur} \quad \Gamma^{-}(\Omega). \tag{1.15}$$

L'équation de transport (1.13) ajoutée aux équations de conservation de la masse (1.2) et du mouvement (1.3) complètent ainsi la description du problème d'écoulement à surface libre.

1.2 Modélisation des écoulements incompressibles : problème discret

La section précédente nous a permis de présenter les équations de modélisation :

$$\begin{cases} \rho\left(\dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}\right) - \boldsymbol{\nabla} \cdot \left(2\mu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\right) + \boldsymbol{\nabla} p &= \rho \boldsymbol{f} \ ; \\ \dfrac{\partial F}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} F &= 0 \ ; \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} &= 0. \end{cases} \tag{1.16}$$

Remarque : les équations de modélisation (1.16) sont définies sur l'ensemble du domaine Ω. L'indexation permettant d'identifier les fluides dans les équations (1.2), (1.3) et (1.13) n'est plus nécessaire car c'est par le biais de l'équation de transport que cette identification va pouvoir s'effectuer.

Les grandes lignes de la résolution du modèle mathématique (1.16) sont :

- Obtention de la forme faible des équations (formulation variationnelle) ;
- Discrétisation de la forme faible ;
- Résolution du problème discret.

Cette section va nous permettre de présenter chacune des trois étapes.

1.2.1 Espaces fonctionnels

Afin d'obtenir la formulation variationnelle des équations de modélisation, nous avons besoin de définir quelques espaces fonctionnels. Soit tout d'abord

$$L^2(\Omega) = \{v : \Omega \to \mathbb{R} \mid \int_\Omega v^2 \, d\Omega < \infty\},$$

l'espace des fonctions de carré intégrable, muni du produit scalaire

$$\left(u, v\right)_{0,\Omega} = \int_\Omega u \, v \, d\Omega.$$

La généralisation en dimension n nous donne :

$$L^2(\Omega)^n = \{\boldsymbol{v} : \Omega \to \mathbb{R}^n \mid \int_\Omega \boldsymbol{v} \cdot \boldsymbol{v} \, d\Omega < \infty\},$$

ainsi que le produit scalaire correspondant :

$$\left(\boldsymbol{u}, \boldsymbol{v}\right)_{0,\Omega} = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} \, d\Omega.$$

Dans le cas des tenseurs, il suffit de remplacer le produit scalaire $\boldsymbol{v} \cdot \boldsymbol{v}$ par la double contraction $\boldsymbol{\tau} : \boldsymbol{\tau} = \sum_{i=1}^{n} \sum_{j=1}^{n} \tau_{ij} \tau_{ji}$ où $\boldsymbol{\tau} : \Omega \to \mathbb{R}^{n \times n}$.

Un autre espace important est l'espace de Sobolev $H^1(\Omega)$:

$$H^1(\Omega) = \{v \in L^2(\Omega) \mid \frac{\partial v}{\partial x_i} \in L^2(\Omega),\ i = 1, \ldots, n\}$$

qui est muni du produit scalaire

$$(u, v)_{1,\Omega} = (u, v)_{0,\Omega} + \sum_{i=1}^{n} (\frac{\partial u}{\partial x_i}, \frac{\partial v}{\partial x_i})_{0,\Omega}.$$

Comme précédemment, la généralisation en dimension n nous donne $H^1(\Omega)^n$ et le produit scalaire correspondant. Un espace dérivé de $H^1(\Omega)^n$, nécessaire à l'étude de la formulation variationnelle de l'équation de la conservation de la quantité de mouvement, est défini comme suit :

$$H^1_{D_u}(\Omega)^n = \{\boldsymbol{v} \in H^1(\Omega)^n \mid \boldsymbol{v} = \mathbf{0} \text{ sur } \Gamma_{D_u}\},$$

c'est-à-dire l'ensemble des fonctions de $H^1(\Omega)^n$ qui sont nulles là où une condition de Dirichlet a été imposée. Finalement, l'espace $H^{\frac{1}{2}}(\Gamma_0)$ est l'ensemble des fonctions définies sur une partie de la frontière de Ω, *i.e.* $\Gamma_0 \subset \partial\Omega$, et formé de la restriction au bord des fonctions de $H^1(\Omega)$. L'espace dual $H^{-\frac{1}{2}}(\Gamma_0)$ est formé des fonctionnelles bornées sur $H^{\frac{1}{2}}(\Gamma_0)$. Pour traiter les problèmes à surface libre, l'espace de base est une variante de l'espace de Sobolev, approprié pour l'étude des problèmes de transport,

$$\Phi_{\boldsymbol{u}}(\Omega) = \{\varphi \in L^2(\Omega) \mid \boldsymbol{u} \cdot \boldsymbol{\nabla}\varphi \in L^2(\Omega) \text{ et } \varphi|_{\Gamma^-} \in L^2_{\boldsymbol{u}}(\Gamma^-)\},$$

où

$$L^2_{\boldsymbol{u}}(\Gamma) = \{\varphi \in L^2(\Gamma) \mid \int_\Gamma |\boldsymbol{u} \cdot \boldsymbol{n}|\varphi^2\, d\Gamma < \infty\}$$

où $\boldsymbol{n}$ est la normale extérieure à Γ. La frontière Γ^- a été définie en (1.14). Le produit scalaire associé à $L^2_{\boldsymbol{u}}$ est le suivant :

$$\langle \varphi, \phi \rangle_{0,\Gamma} = \int_\Gamma (\boldsymbol{u} \cdot \boldsymbol{n})\, \varphi\, \phi\, d\Gamma.$$

Pour exprimer la forme faible de l'équation de tranport, nous ferons appel à l'espace

$$\Phi_{\boldsymbol{u},\Gamma^-}(\Omega) = \{\varphi \in \Phi_{\boldsymbol{u}}(\Omega) \mid \varphi = 0 \text{ sur } \Gamma^-\}.$$

Les inconnues $\boldsymbol{u}$, p et F appartiennent respectivement aux espaces fonctionnels $H_1(\Omega)^n$, $L^2(\Omega)$ et $\Phi_{\boldsymbol{u}}(\Omega)$.

1.2.2 Formulation variationnelle des équations

On présente dans cette section l'une des composantes importantes de la théorie des éléments finis, la *formulation variationnelle*. On obtient par le biais de cette méthode la *forme faible* des équations de modélisation (1.2), (1.3) et (1.13). Cette forme faible s'obtient en deux étapes :

1. multiplier les équations par des fonctions test appropriées ;
2. intégrer par parties certains termes sur le domaine de calcul Ω.

Ce procédé permet de réduire l'ordre des dérivées et de faire apparaître les conditions aux frontières naturelles.

1.2.2.1 Équation de conservation de la quantité de mouvement

Nous supposons dans cette section que $\boldsymbol{f}$ est élément de $L^2(\Omega)^n$. Voici donc les étapes qui mènent à la forme faible de l'équation de conservation de la quantité de mouvement.

Première étape

Nous rappelons que l'équation de conservation de la quantité de mouvement est la suivante :

$$\rho\Big(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\Big) - \boldsymbol{\nabla}\cdot\big(2\mu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big) + \boldsymbol{\nabla}p = \rho\boldsymbol{f}. \tag{1.17}$$

En multipliant (1.17) par la fonction test $\boldsymbol{w} \in H^1_{D_{\boldsymbol{u}}}(\Omega)^n$ et en intégrant sur le domaine Ω, nous obtenons :

$$\int_\Omega \left(\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho(\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\right)\cdot\boldsymbol{w}\, d\Omega - \int_\Omega \boldsymbol{\nabla}\cdot\left(2\mu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\right)\cdot\boldsymbol{w}\, d\Omega + \int_\Omega \boldsymbol{\nabla}p\cdot\boldsymbol{w}\, d\Omega = \int_\Omega \rho\boldsymbol{f}\cdot\boldsymbol{w}\, d\Omega. \quad (1.18)$$

Deuxième étape

Procédons à présent à l'intégration par parties du deuxième et du troisième terme du membre de gauche de l'équation (1.18). Tout d'abord, nous avons pour le terme en pression :

$$\boldsymbol{\nabla}\cdot(p\boldsymbol{w}) = \boldsymbol{\nabla}p\cdot\boldsymbol{w} + p(\boldsymbol{\nabla}\cdot\boldsymbol{w})$$

$$\int_\Omega \boldsymbol{\nabla}\cdot(p\boldsymbol{w})\, d\Omega = \int_\Omega \boldsymbol{\nabla}p\cdot\boldsymbol{w}\, d\Omega + \int_\Omega p\boldsymbol{\nabla}\cdot\boldsymbol{w}\, d\Omega. \quad (1.19)$$

Le théorème de la divergence impose que le flux d'un champ vectoriel sur la frontière Γ doit être égal à la divergence de ce champ sur le domaine Ω. Appliqué au membre de gauche de l'équation (1.19), on obtient :

$$\int_\Omega \boldsymbol{\nabla}\cdot(p\boldsymbol{w})\, d\Omega = \int_{\Gamma=\partial\Omega} (p\boldsymbol{w})\cdot\boldsymbol{n}\, d\Gamma.$$

Remarquons de plus que l'intégrant de droite possède une deuxième forme :

$$(p\boldsymbol{w})\cdot\boldsymbol{n} = (p\boldsymbol{I}\cdot\boldsymbol{n})\cdot\boldsymbol{w}.$$

La relation (1.19) devient alors :

$$\int_\Gamma (p\boldsymbol{I}\cdot\boldsymbol{n})\cdot\boldsymbol{w}\, d\Gamma = \int_\Omega \boldsymbol{\nabla}p\cdot\boldsymbol{w}\, d\Omega + \int_\Omega p\boldsymbol{\nabla}\cdot\boldsymbol{w}\, d\Omega. \quad (1.20)$$

Pour le terme de diffusion visqueuse, nous avons :

$$\boldsymbol{\nabla}\cdot\left(\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\cdot\boldsymbol{w}\right) = \left(\boldsymbol{\nabla}\cdot\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\right)\cdot\boldsymbol{w} + \dot{\boldsymbol{\gamma}}(\boldsymbol{u}) : \boldsymbol{\nabla}\boldsymbol{w}$$

$$\Leftrightarrow$$

$$\int_\Omega \boldsymbol{\nabla}\cdot\left(\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\cdot\boldsymbol{w}\right) d\Omega = \int_\Omega \left(\boldsymbol{\nabla}\cdot\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\right)\cdot\boldsymbol{w}\, d\Omega + \int_\Omega \dot{\boldsymbol{\gamma}}(\boldsymbol{u}) : \boldsymbol{\nabla}\boldsymbol{w}\, d\Omega.$$

En utilisant à nouveau le théorème de la divergence, nous avons :

$$\int_\Omega \boldsymbol{\nabla} \cdot \big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}) \cdot \boldsymbol{w}\big)\, d\Omega = \int_\Gamma \big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}) \cdot \boldsymbol{w}\big) \cdot \boldsymbol{n}\, d\Gamma.$$

Ainsi,

$$\int_\Gamma \big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}) \cdot \boldsymbol{w}\big) \cdot \boldsymbol{n}\, d\Gamma = \int_\Omega \big(\boldsymbol{\nabla} \cdot \dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big) \cdot \boldsymbol{w}\, d\Omega + \int_\Omega \dot{\boldsymbol{\gamma}}(\boldsymbol{u}) : \boldsymbol{\nabla}\boldsymbol{w}\, d\Omega.$$

Remarquons que le dernier terme de cette équation est le produit de deux tenseurs dont le premier est symétrique, ce qui nous permet d'écrire :

$$\begin{aligned}
\dot{\boldsymbol{\gamma}}(\boldsymbol{u}) : \boldsymbol{\nabla}\boldsymbol{w} &= \tfrac{1}{2}\big(\boldsymbol{\nabla}\boldsymbol{u} + (\boldsymbol{\nabla}\boldsymbol{u})^T\big) : \boldsymbol{\nabla}\boldsymbol{w} \\
&= \tfrac{1}{2}\big(\boldsymbol{\nabla}\boldsymbol{u} + (\boldsymbol{\nabla}\boldsymbol{u})^T\big) : (\boldsymbol{\nabla}\boldsymbol{w})^T \\
&= \tfrac{1}{2}\big(\boldsymbol{\nabla}\boldsymbol{u} + (\boldsymbol{\nabla}\boldsymbol{u})^T\big) : \tfrac{1}{2}\big(\boldsymbol{\nabla}\boldsymbol{w} + (\boldsymbol{\nabla}\boldsymbol{w})^T\big) \\
&= \dot{\boldsymbol{\gamma}}(\boldsymbol{u}) : \dot{\boldsymbol{\gamma}}(\boldsymbol{w}).
\end{aligned}$$

Finalement, nous obtenons :

$$-\int_\Omega \big(\boldsymbol{\nabla} \cdot \dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big) \cdot \boldsymbol{w}\, d\Omega = \int_\Omega \dot{\boldsymbol{\gamma}}(\boldsymbol{u}) : \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\, d\Omega - \int_\Gamma \big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}) \cdot \boldsymbol{n}\big) \cdot \boldsymbol{w}\, d\Gamma. \tag{1.21}$$

La combinaison des intégrations par parties des termes de pression et de diffusion visqueuse (équations (1.20) et (1.21)) nous permet d'obtenir la forme faible de l'équation de conservation de la quantité de mouvement :

$$\begin{aligned}
\rho\big(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}, \boldsymbol{w}\big)_{0,\Omega} + 2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega} - \big(p, \boldsymbol{\nabla} \cdot \boldsymbol{w}\big)_{0,\Omega} = \\
\langle \boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{w} \rangle_{H^{-\frac{1}{2}}(\Gamma)^n, H^{\frac{1}{2}}(\Gamma)^n} + \rho\big(\boldsymbol{f}, \boldsymbol{w}\big)_{0,\Omega}.
\end{aligned} \tag{1.22}$$

Nous avons introduit ici la notation $\langle \boldsymbol{a}, \boldsymbol{b} \rangle_{H^{-\frac{1}{2}}(\Gamma), H^{\frac{1}{2}}(\Gamma)}$ qui indique que $\boldsymbol{a} \in H^{-\frac{1}{2}}(\Gamma)$ et $\boldsymbol{b} \in H^{\frac{1}{2}}(\Gamma)$. Rappelons pour finir que la frontière de Ω se divise en deux parties $\Gamma = \Gamma_{D_{\boldsymbol{u}}} \cup \Gamma_{N_{\boldsymbol{u}}}$ et que les fonctions test $\boldsymbol{w}$ s'annulent sur la partie $\Gamma_{D_{\boldsymbol{u}}}$ où ont été imposées les conditions de Dirichlet, de sorte que

$$\begin{aligned}
\langle \boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{w} \rangle_{H^{-\frac{1}{2}}(\Gamma)^n, H^{\frac{1}{2}}(\Gamma)^n} &= \langle \boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{w} \rangle_{H^{-\frac{1}{2}}(\Gamma_{N_{\boldsymbol{u}}})^n, H^{\frac{1}{2}}(\Gamma_{N_{\boldsymbol{u}}})^n} \\
&= \langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w} \rangle_{H^{-\frac{1}{2}}(\Gamma_{N_{\boldsymbol{u}}})^n, H^{\frac{1}{2}}(\Gamma_{N_{\boldsymbol{u}}})^n},
\end{aligned}$$

où $\boldsymbol{t}_{N_{\boldsymbol{u}}}$ a été définie en (1.4). Afin de ne pas alourdir inutilement les équations, nous noterons simplement

$$\langle \boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{w} \rangle_{H^{-\frac{1}{2}}(\Gamma_{N_{\boldsymbol{u}}})^n, H^{\frac{1}{2}}(\Gamma_{N_{\boldsymbol{u}}})^n} = \langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w} \rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}}.$$

1.2.2.2 Équation de transport de la pseudo-concentration

Première étape

En multipliant l'équation de transport par la fonction test $\varphi \in \Phi_{\boldsymbol{u},\Gamma^-}(\Omega)$, nous obtenons :

$$\int_\Omega \frac{\partial F}{\partial t}\varphi \, d\Omega + \int_\Omega (\boldsymbol{u} \cdot \boldsymbol{\nabla} F)\varphi \, d\Omega = 0 \Leftrightarrow \left(\frac{\partial F}{\partial t}, \varphi\right)_{0,\Omega} + \left(\boldsymbol{u} \cdot \boldsymbol{\nabla} F, \varphi\right)_{0,\Omega} = 0. \quad (1.23)$$

Deuxième étape

Pas d'intégration par parties nécessaire.

Remarque : L'équation de transport de la pseudo-concentration, de par sa nature hyperbolique, requiert une attention particulière. En effet, sa résolution par la méthode de Galerkin (cf. section 1.2.3) peut mener à d'importantes instabilités numériques se traduisant par des oscillations non physiques dans la solution (Johnson 1990). Une approche possible consiste à introduire une composante elliptique à l'équation (1.13) en rajoutant un terme de diffusion artificielle qui n'agit que dans le sens des lignes de courant. Parmis les méthodes les plus populaires, citons la méthode « streamline upwind / Petrov-Galerkin » (SUPG) (Brooks and Hughes 1990) qui modifie la fonction test de sorte que la forme faible (1.23) devient :

$$\left(\frac{\partial F}{\partial t}, \bar{\varphi}\right)_{0,\Omega} + \left(\boldsymbol{u} \cdot \boldsymbol{\nabla} F, \bar{\varphi}\right)_{0,\Omega} = 0, \tag{1.24}$$

où

$$\bar{\varphi} = \varphi + \beta \boldsymbol{u} \cdot \boldsymbol{\nabla} \varphi.$$

avec β un paramètre de stabilisation. On comprend mieux les implications de ce choix lorsque l'on exprime le deuxième terme de l'équation (1.24) comme suit :

$$\begin{aligned} \left(\boldsymbol{u} \cdot \boldsymbol{\nabla} F, \bar{\varphi}\right)_{0,\Omega} &= \left(\boldsymbol{u} \cdot \boldsymbol{\nabla} F, \varphi\right)_{0,\Omega} + \beta \left(\boldsymbol{u} \cdot \boldsymbol{\nabla} F, \boldsymbol{u} \cdot \boldsymbol{\nabla} \varphi\right)_{0,\Omega} \\ &= \left(\boldsymbol{u} \cdot \boldsymbol{\nabla} F, \varphi\right)_{0,\Omega} + \beta \left(\boldsymbol{u}\boldsymbol{u}^T \boldsymbol{\nabla} F, \boldsymbol{\nabla} \varphi\right)_{0,\Omega}. \end{aligned}$$

Le tenseur $\beta \boldsymbol{u}\boldsymbol{u}^T$ est le terme de diffusion artificielle dans le sens des lignes de courant. En pratique, on retrouve plutôt la formulation SUPG exprimée sous la forme

$$\left(\frac{\partial F}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} F, \varphi\right)_{0,\Omega} + \sum_K \int_K \left(\frac{\partial F}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} F\right)\left(\beta \boldsymbol{u} \cdot \boldsymbol{\nabla} \varphi\right) d\boldsymbol{x}, \tag{1.25}$$

où K est un élément du maillage. On retrouve la forme faible de la méthode de Galerkin, plus la partie stabilisée. Dans la suite de ce mémoire, par soucis de simplicité et cela ne changeant pas fondamentalement la structure matricielle que nous allons obtenir, nous ne considérons que l'équation de transport non stabilisée même si l'utilisation de techniques de stabilisation est primordial en pratique.

1.2.2.3 Équation de conservation de la masse

Première étape

La fonction test q est choisie dans l'espace fonctionnel $L^2(\Omega)$, de sorte que :

$$\int_\Omega q\boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 \Leftrightarrow \left(q, \boldsymbol{\nabla}\cdot\boldsymbol{u}\right)_{0,\Omega} = 0. \tag{1.26}$$

Deuxième étape

Pas d'intégration par parties nécessaire.

Les expressions (1.22), (1.23) et (1.26) représentent ainsi la forme faible du système d'équations (1.16). Le tableau 1.1 récapitule les nouvelles équations obtenues par le passage du problème fort à la forme faible. Nous pouvons donc passer à l'étape suivante qui est celle de la discrétisation de la forme faible.

1.2.3 Discrétisation des équations : méthode de Galerkin

Une fois le modèle mathématique mis en place et la formulation faible obtenue, il nous reste à discrétiser ces équations. Cette étape nécessite l'utilisation de fonctions d'interpolation. Ainsi, chacune des variables u, v, p et F s'exprime comme une combinaison linéaire de ces fonctions d'interpolation :

$$\begin{aligned} u \approx u_h &= \sum_{j=1}^{N^u} u_j(t)\phi_j^u(\boldsymbol{x}), & v \approx v_h &= \sum_{j=1}^{N^v} v_j(t)\phi_j^v(\boldsymbol{x}) ; \\ p \approx p_h &= \sum_{j=1}^{N^p} p_j(t)\phi_j^p(\boldsymbol{x}), & F \approx F_h &= \sum_{j=1}^{N^F} F_j(t)\phi_j^F(\boldsymbol{x}), \end{aligned}$$

où ϕ_j^u et ϕ_j^v appartiennent à l'espace fonctionnel discret $V_h \subset H^1(\Omega)$, ϕ_j^p à l'espace $Q_h \subset L^2(\Omega)$ et ϕ_j^F à l'espace $\Phi_{\boldsymbol{u},h} \subset \Phi_{\boldsymbol{u}}(\Omega)$. Si on note α l'une des variables u, v, p ou F, on notera alors α_h l'approximation éléments-finis de α, ϕ_j^α la j^e fonction d'interpolation associée à la variable α et N^α le nombre de fonctions d'interpolation

Tableau 1.1 – Obtention de la forme faible - Résumé

Le problème fort

$$
\begin{aligned}
&(1)\quad \rho\Big(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\Big) - \boldsymbol{\nabla}\cdot\big(2\mu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big) + \boldsymbol{\nabla}p &&= \rho\boldsymbol{f}\,;\\
&(2)\quad \frac{\partial F}{\partial t} + \boldsymbol{u}\cdot\boldsymbol{\nabla}F &&= 0\,;\\
&(3)\quad \boldsymbol{\nabla}\cdot\boldsymbol{u} &&= 0.
\end{aligned}
$$

La forme faible

$$
\begin{aligned}
(1)\Rightarrow\quad & \underbrace{\rho\Big(\frac{\partial \boldsymbol{u}}{\partial t},\boldsymbol{w}\Big)_{0,\Omega}}_{\text{Terme transitoire}} + \underbrace{\rho\big((\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u},\boldsymbol{w}\big)_{0,\Omega}}_{\text{Convection}} + \underbrace{2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}}_{\text{Diffusion}} -\\
& \underbrace{\big(p,\boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega}}_{\text{Pression}} = \underbrace{\langle \boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}}_{\substack{\text{Terme}\\ \text{de bord}}} + \underbrace{\rho\big(\boldsymbol{f},\boldsymbol{w}\big)_{0,\Omega}}_{\substack{\text{Force}\\ \text{volumique}}}\,;\\
(2)\Rightarrow\quad & \underbrace{\big(\boldsymbol{u}\cdot\boldsymbol{\nabla}F,\varphi\big)_{0,\Omega}}_{\substack{\text{Transport de la}\\ \text{pseudo-concentration}}} = 0\,;\\
(3)\Rightarrow\quad & \underbrace{\big(\boldsymbol{\nabla}\cdot\boldsymbol{u},q\big)_{0,\Omega}}_{\text{Divergence}} = 0.
\end{aligned}
$$

associées à la variable α. Les fonctions d'interpolation ϕ_j^α sont également choisies pour être les fonctions test qui apparaissent dans la forme faible des équations. C'est la méthode de *Galerkin*. Notons également que la dépendance temporelle apparaît dans les degrés de liberté u_j, v_j, p_j et F_j et que la dépendance spatiale apparaît dans les fonctions d'interpolation ϕ_j^u, ϕ_j^v, ϕ_j^p et ϕ_j^F. Explicitons la discrétisation de chacune des équations de modélisation.

L'équation de conservation de la quantité de mouvement

Nous discrétisons dans cette section l'équation (1.22) dans le cas bidimensionnel (en x et en y). Occupons-nous d'abord du terme transitoire :

$$\rho\big(\frac{\partial \boldsymbol{u}}{\partial t}, \boldsymbol{w}\big)_{0,\Omega} \approx \rho\big(\frac{\partial \boldsymbol{u}_h}{\partial t}, \boldsymbol{w}\big)_{0,\Omega}.$$

En choisissant successivement $\boldsymbol{w} = \big(\phi_i^u, 0\big)$ pour chaque $i = 1, \ldots, N^u$, on obtient :

$$\rho\big(\frac{\partial \boldsymbol{u}_h}{\partial t}, \boldsymbol{w}\big)_{0,\Omega} = \sum_{j=1}^{N^u} \frac{\partial u_j}{\partial t} \rho\big(\phi_j^u, \phi_i^u\big)_{0,\Omega}, \quad i = 1, \ldots N^u, \tag{1.27}$$

et en choisissant successivement $\boldsymbol{w} = \big(0, \phi_i^v\big)$ pour chaque $i = 1, \ldots, N^v$, on obtient :

$$\rho\big(\frac{\partial \boldsymbol{u}_h}{\partial t}, \boldsymbol{w}\big)_{0,\Omega} = \sum_{j=1}^{N^v} \frac{\partial v_j}{\partial t} \rho\big(\phi_j^v, \phi_i^v\big)_{0,\Omega}, \quad i = 1, \ldots N^v. \tag{1.28}$$

Sous forme matricielle, les relations (1.27) et (1.28) s'écrivent :

$$\begin{pmatrix} T^u & 0 \\ 0 & T^v \end{pmatrix} \begin{pmatrix} \dot{\vec{U}} \\ \dot{\vec{V}} \end{pmatrix}, \tag{1.29}$$

où

$$\begin{cases} T_{ij}^u = \rho\big(\phi_j^u, \phi_i^u\big)_{0,\Omega} \quad , \quad T_{ij}^v = \rho\big(\phi_j^v, \phi_i^v\big)_{0,\Omega} ; \\ \dot{\vec{U}}_j = \dfrac{\partial u_j}{\partial t} \qquad\qquad , \quad \dot{\vec{V}}_j = \dfrac{\partial v_j}{\partial t}. \end{cases}$$

Pour le terme de diffusion visqueuse, on obtient :

$$2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega} \approx 2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}_h), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}.$$

Comme précédemment, on considère dans un premier temps successivement les fonctions test $\boldsymbol{w} = \left(\phi_i^u, 0\right)$, $i = 1, \ldots, N^u$. En notant $\boldsymbol{u}_h = (u_h \ v_h)^T$, on obtient :

$$
\begin{aligned}
&\left(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}_h), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\right)_{0,\Omega} = \\
&= \left(\begin{pmatrix} \dfrac{\partial u_h}{\partial x} & \dfrac{1}{2}\left(\dfrac{\partial u_h}{\partial y} + \dfrac{\partial v_h}{\partial x}\right) \\ \dfrac{1}{2}\left(\dfrac{\partial u_h}{\partial y} + \dfrac{\partial v_h}{\partial x}\right) & \dfrac{\partial v_h}{\partial y} \end{pmatrix} : \begin{pmatrix} \dfrac{\partial \phi_i^u}{\partial x} & \dfrac{1}{2}\dfrac{\partial \phi_i^u}{\partial y} \\ \dfrac{1}{2}\dfrac{\partial \phi_i^u}{\partial y} & 0 \end{pmatrix} \right)_{0,\Omega} \\
&= \left(\frac{\partial u_h}{\partial x}, \frac{\partial \phi_i^u}{\partial x}\right)_{0,\Omega} + \frac{1}{2}\left(\frac{\partial u_h}{\partial y} + \frac{\partial v_h}{\partial x}, \frac{\partial \phi_i^u}{\partial y}\right)_{0,\Omega} \\
&= \sum_{j=1}^{N^u} u_j \left(\frac{\partial \phi_j^u}{\partial x}, \frac{\partial \phi_i^u}{\partial x}\right)_{0,\Omega} + \frac{1}{2}\left(\sum_{j=1}^{N^u} u_j \frac{\partial \phi_j^u}{\partial y} + \sum_{j=1}^{N^v} v_j \frac{\partial \phi_j^v}{\partial x}, \frac{\partial \phi_i^v}{\partial y}\right)_{0,\Omega} \\
&= \sum_{j=1}^{N^u} u_j \left[\left(\frac{\partial \phi_j^u}{\partial x}, \frac{\partial \phi_i^u}{\partial x}\right)_{0,\Omega} + \frac{1}{2}\left(\frac{\partial \phi_j^u}{\partial y}, \frac{\partial \phi_i^u}{\partial y}\right)_{0,\Omega}\right] + \sum_{j=1}^{N^v} v_j \left[\frac{1}{2}\left(\frac{\partial \phi_j^v}{\partial x}, \frac{\partial \phi_i^v}{\partial y}\right)_{0,\Omega}\right], \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 1, \ldots, N^u.
\end{aligned}
$$

Dans un second temps, on considère successivement les fonctions test $\boldsymbol{w} = \left(0, \phi_i^v\right)$, $i = 1, \ldots N^v$:

$$
\begin{aligned}
&\left(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}_h), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\right)_{0,\Omega} = \\
&= \sum_{j=1}^{N^v} v_j \left(\frac{\partial \phi_j^v}{\partial y}, \frac{\partial \phi_i^v}{\partial y}\right)_{0,\Omega} + \frac{1}{2}\left(\sum_{j=1}^{N^u} u_j \frac{\partial \phi_j^u}{\partial y} + \sum_{j=1}^{N^v} v_j \frac{\partial \phi_j^v}{\partial x}, \frac{\partial \phi_i^v}{\partial x}\right)_{0,\Omega} \\
&= \sum_{j=1}^{N^u} u_j \left[\frac{1}{2}\left(\frac{\partial \phi_j^u}{\partial y}, \frac{\partial \phi_i^v}{\partial x}\right)_{0,\Omega}\right] + \sum_{j=1}^{N^v} v_j \left[\left(\frac{\partial \phi_j^v}{\partial y}, \frac{\partial \phi_i^v}{\partial y}\right)_{0,\Omega} + \frac{1}{2}\left(\frac{\partial \phi_j^v}{\partial x}, \frac{\partial \phi_i^v}{\partial x}\right)_{0,\Omega}\right], \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 1, \ldots, N^v,
\end{aligned}
$$

soit sous forme matricielle :

$$
\begin{pmatrix} A^{uu} & A^{uv} \\ A^{vu} & A^{vv} \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{V} \end{pmatrix}, \tag{1.30}
$$

où

$$\begin{cases} A_{ij}^{uu} = 2\mu\big(\dfrac{\partial\phi_j^u}{\partial x}, \dfrac{\partial\phi_i^u}{\partial x}\big)_{0,\Omega} + \mu\big(\dfrac{\partial\phi_j^u}{\partial y}, \dfrac{\partial\phi_i^u}{\partial y}\big)_{0,\Omega} ; \\ A_{ij}^{uv} = \mu\big(\dfrac{\partial\phi_j^u}{\partial x}, \dfrac{\partial\phi_i^v}{\partial y}\big)_{0,\Omega} = A_{ji}^{vu} ; \\ A_{ij}^{vv} = 2\mu\big(\dfrac{\partial\phi_j^v}{\partial y}, \dfrac{\partial\phi_i^v}{\partial y}\big)_{0,\Omega} + \mu\big(\dfrac{\partial\phi_j^v}{\partial x}, \dfrac{\partial\phi_i^v}{\partial x}\big)_{0,\Omega}, \end{cases}$$

et où $\vec{U}$ et $\vec{V}$ contiennent respectivement les degrés de liberté des variables u et v.

Quant au terme en pression, on obtient pour $\boldsymbol{w} = \big(\phi_i^u, 0\big)$, $i = 1, \ldots, N^u$:

$$\big(p, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega} \approx \big(p_h, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega} = \sum_{j=1}^{N^p} p_j \big(\phi_j^p, \frac{\partial\phi_i^u}{\partial x}\big)_{0,\Omega},$$

et pour $\boldsymbol{w} = \big(0, \phi_i^v\big)$, $i = 1, \ldots, N^v$:

$$\big(p, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega} \approx \big(p_h, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega} = \sum_{j=1}^{N^p} p_j \big(\phi_j^p, \frac{\partial\phi_i^v}{\partial y}\big)_{0,\Omega},$$

soit sous forme matricielle :

$$\begin{pmatrix} (B^u)^T \\ (B^v)^T \end{pmatrix} \vec{P}, \tag{1.31}$$

où

$$\begin{cases} B_{ji}^u = \big(\phi_j^p, \dfrac{\partial\phi_i^u}{\partial x}\big)_{0,\Omega} ; \\ B_{ji}^v = \big(\phi_j^p, \dfrac{\partial\phi_i^v}{\partial y}\big)_{0,\Omega} \end{cases}$$

et où $\vec{P}$ contient les degrés de liberté de la variable p.

Regardons à présent ce que donne le terme de convection. Pour $\boldsymbol{w} = \big(\phi_i^u, 0\big)$, $i = 1, \ldots, N^u$, on obtient :

$$\rho\big((\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}, \boldsymbol{w}\big)_{0,\Omega} \approx \rho\big((\boldsymbol{u}_h\cdot\boldsymbol{\nabla})\boldsymbol{u}_h, \boldsymbol{w}\big)_{0,\Omega} = \rho(\boldsymbol{u}_h\cdot\boldsymbol{\nabla}) \sum_{j=1}^{N^u} u_j \big(\phi_j^u, \phi_i^u\big)_{0,\Omega},$$

et pour $\boldsymbol{w} = \big(0, \phi_i^v\big)$, $i = 1, \ldots, N^v$:

$$\rho\big((\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}, \boldsymbol{w}\big)_{0,\Omega} \approx \rho\big((\boldsymbol{u}_h\cdot\boldsymbol{\nabla})\boldsymbol{u}_h, \boldsymbol{w}\big)_{0,\Omega} = \rho(\boldsymbol{u}_h\cdot\boldsymbol{\nabla})\sum_{j=1}^{N^v} v_j\big(\phi_j^v, \phi_i^v\big)_{0,\Omega},$$

ce qui donne sous forme matricielle :

$$\begin{pmatrix} C^u(\boldsymbol{u}_h) & 0 \\ 0 & C^v(\boldsymbol{u}_h) \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{V} \end{pmatrix}, \tag{1.32}$$

où

$$\begin{cases} C^u(\boldsymbol{u}_h)_{ij} &= \rho\big(\boldsymbol{u}_h\cdot\boldsymbol{\nabla}\phi_j^u, \phi_i^u\big)_{0,\Omega}\,; \\ C^v(\boldsymbol{u}_h)_{ij} &= \rho\big(\boldsymbol{u}_h\cdot\boldsymbol{\nabla}\phi_j^v, \phi_i^v\big)_{0,\Omega}. \end{cases}$$

Remarque :

Les matrices de la forme $C^u(\boldsymbol{u}_h)$ sont des abus de notation qui nous permettent de mettre en évidence la dépendance en $\boldsymbol{u}_h$ de ces termes non linéaires et de simplifier la présentation de la formulation variationnelle. Nous cesserons d'utiliser cette notation, une fois que l'étape de linéarisation sera effectuée (cf. section 1.2.4).

Enfin, il nous reste à voir ce que donne le membre de droite de l'équation (1.22). Pour $\boldsymbol{w} = \big(\phi_i^u, 0\big)$, $i = 1, \ldots, N^u$, on obtient :

$$\langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w}\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} + \rho\big(\boldsymbol{f}, \boldsymbol{w}\big)_{0,\Omega} = \langle t_x, \phi_i^u\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} + \rho\big(f_x, \phi_i^u\big)_{0,\Omega},$$

et pour $\boldsymbol{w} = \big(0, \phi_i^v\big)$, $i = 1, \ldots, N^v$:

$$\langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w}\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} + \rho\big(\boldsymbol{f}, \boldsymbol{w}\big)_{0,\Omega} = \langle t_y, \phi_i^v\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} + \rho\big(f_y, \phi_i^v\big)_{0,\Omega}.$$

La forme matricielle finale de la discrétisation éléments-finis de l'équation de mouvement est donc :

$$\begin{pmatrix} T^u & 0 \\ 0 & T^v \end{pmatrix} \begin{pmatrix} \dot{\vec{U}} \\ \dot{\vec{V}} \end{pmatrix} + \begin{pmatrix} C^u(\boldsymbol{u}_h) + A^{uu} & A^{uv} \\ A^{vu} & C^v(\boldsymbol{u}_h) + A^{vv} \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{V} \end{pmatrix} + \begin{pmatrix} (B^u)^T \\ (B^v)^T \end{pmatrix} \vec{P} = \begin{pmatrix} \vec{E}^u \\ \vec{E}^v \end{pmatrix}, \tag{1.33}$$

où

$$\vec{E}_i^u = \langle t_x, \phi_i^u \rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} + \rho(f_x, \phi_i^u)_{0,\Omega} \quad \text{et} \quad \vec{E}_i^v = \langle t_y, \phi_i^v \rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} + \rho(f_y, \phi_i^v)_{0,\Omega}.$$

L'équation de transport de la pseudo-concentration

Nous discrétisons ici la forme faible de l'équation de transport (1.23) :

$$(\frac{\partial F}{\partial t}, \varphi)_{0,\Omega} + (\boldsymbol{u} \cdot \boldsymbol{\nabla} F, \varphi)_{0,\Omega} \approx (\frac{\partial F_h}{\partial t}, \varphi)_{0,\Omega} + (\boldsymbol{u}_h \cdot \boldsymbol{\nabla} F_h, \varphi)_{0,\Omega}.$$

Les fonctions test φ sont les fonctions d'interpolation $\varphi = \phi_i^F$, $i = 1, \ldots, N^F$, ce qui nous donne :

$$\begin{aligned}
&(\frac{\partial F_h}{\partial t}, \varphi)_{0,\Omega} + (\boldsymbol{u}_h \cdot \boldsymbol{\nabla} F_h, \varphi)_{0,\Omega} = \\
&= \sum_{j=1}^{N^F} \frac{\partial F_j}{\partial t} (\phi_j^F, \phi_i^F)_{0,\Omega} + \sum_{j=1}^{N^F} F_j (u_h \frac{\partial \phi_j^F}{\partial x} + v_h \frac{\partial \phi_j^F}{\partial y}, \phi_i^F)_{0,\Omega}.
\end{aligned}$$

Sous forme matricielle, nous obtenons :

$$T^F \dot{\vec{F}} + K(\boldsymbol{u}_h)\vec{F} = \vec{0}, \tag{1.34}$$

où

$$\begin{cases} K_{ij}(\boldsymbol{u}_h) &= (u_h \frac{\partial \phi_j^F}{\partial x} + v_h \frac{\partial \phi_j^F}{\partial y}, \phi_i^F)_{0,\Omega}\,; \\ T^F &= (\phi_j^F, \phi_i^F)_{0,\Omega}, \end{cases}$$

et où $\vec{F}$ et $\dot{\vec{F}}$ contiennent respectivement les degrés de liberté des variables F et $\frac{\partial F}{\partial t}$.

L'équation de conservation de la masse

Nous discrétisons ici la forme faible de l'équation de conservation de la masse (1.26).

On obtient :

$$\begin{aligned}(\boldsymbol{\nabla}\cdot\boldsymbol{u},q)_{0,\Omega} &\approx (\boldsymbol{\nabla}\cdot\boldsymbol{u}_h,q)_{0,\Omega}\\ &\approx (\frac{\partial}{\partial x}\sum_{j=1}^{N^u}u_j\phi_j^u+\frac{\partial}{\partial y}\sum_{j=1}^{N^v}v_j\phi_j^v,q)_{0,\Omega}\\ &\approx \sum_{j=1}^{N^u}u_j(\frac{\partial\phi_j^u}{\partial x},q)_{0,\Omega}+\sum_{j=1}^{N^v}v_j(\frac{\partial\phi_j^v}{\partial y},q)_{0,\Omega}.\end{aligned}$$

On prend comme fonction test q les fonctions d'interpolation ϕ_i^p, $i=1,\dots N^p$. Ainsi,

$$(\boldsymbol{\nabla}\cdot\boldsymbol{u},q)_{0,\Omega}\approx\sum_{j=1}^{N^u}u_j(\frac{\partial\phi_j^u}{\partial x},\phi_i^p)_{0,\Omega}+\sum_{j=1}^{N^v}v_j(\frac{\partial\phi_j^v}{\partial y},\phi_i^p)_{0,\Omega},\quad i=1,\dots,N^p,$$

ce qui se traduit sous forme matricielle par :

$$\begin{pmatrix}B^u & B^v\end{pmatrix}\begin{pmatrix}\vec{U}\\ \vec{V}\end{pmatrix}=\vec{0}, \tag{1.35}$$

où

$$\begin{cases}B_{ij}^u &= (\phi_i^p,\dfrac{\partial\phi_j^u}{\partial x})_{0,\Omega}\,;\\ B_{ij}^v &= (\phi_i^p,\dfrac{\partial\phi_j^v}{\partial y})_{0,\Omega}.\end{cases}$$

Nous pouvons à présent additionner les systèmes (1.33), (1.34) et (1.35) pour obtenir le système matriciel global à résoudre :

$$\begin{pmatrix}T^u & 0 & 0 & 0\\ 0 & T^v & 0 & 0\\ 0 & 0 & T^F & 0\\ 0 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\dot{\vec{U}}\\ \dot{\vec{V}}\\ \dot{\vec{F}}\\ \dot{\vec{P}}\end{pmatrix}+$$

$$\begin{pmatrix}A^{uu}+C^u(\boldsymbol{u}_h) & A^{uv} & 0 & (B^u)^T\\ A^{vu} & A^{vv}+C^v(\boldsymbol{u}_h) & 0 & (B^v)^T\\ 0 & 0 & K(\boldsymbol{u}_h) & 0\\ B^u & B^v & 0 & 0\end{pmatrix}\begin{pmatrix}\vec{U}\\ \vec{V}\\ \vec{F}\\ \vec{P}\end{pmatrix}=\begin{pmatrix}\vec{E}^u\\ \vec{E}^v\\ \vec{0}\\ \vec{0}\end{pmatrix}. \tag{1.36}$$

Tableau 1.2 – Discrétisation des équations de modélisation - Résumé

La forme faible

$$\begin{cases} \rho\left(\frac{\partial \boldsymbol{u}}{\partial t}, \boldsymbol{w}\right)_{0,\Omega} + \rho\left((\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}, \boldsymbol{w}\right)_{0,\Omega} + 2\mu\left(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\right)_{0,\Omega} - \\ \left(p, \boldsymbol{\nabla}\cdot\boldsymbol{w}\right)_{0,\Omega} = \langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w}\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} + \rho\left(\boldsymbol{f}, \boldsymbol{w}\right)_{0,\Omega}; \\ \left(\boldsymbol{u}\cdot\boldsymbol{\nabla}F, \varphi\right)_{0,\Omega} = 0; \\ \left(\boldsymbol{\nabla}\cdot\boldsymbol{u}, q\right)_{0,\Omega} = 0. \end{cases}$$

La discrétisation par la méthode de Galerkin

$$\begin{pmatrix} T^u & 0 & 0 & 0 \\ 0 & T^v & 0 & 0 \\ 0 & 0 & T^F & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{\vec{U}} \\ \dot{\vec{V}} \\ \dot{\vec{F}} \\ \dot{\vec{P}} \end{pmatrix} +$$

$$\begin{pmatrix} A^{uu} + C^u(\boldsymbol{u}_h) & A^{uv} & 0 & (B^u)^T \\ A^{vu} & A^{vu} + C^v(\boldsymbol{u}_h) & 0 & (B^v)^T \\ 0 & 0 & K(\boldsymbol{u}_h) & 0 \\ B^u & B^v & 0 & 0 \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{V} \\ \vec{F} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{E}^u \\ \vec{E}^v \\ \vec{0} \\ \vec{0} \end{pmatrix}.$$

Le tableau 1.2 présente un résumé des principales étapes par lesquelles nous sommes passées dans les sections précédentes. Le système (1.36) est clairement non linéaire, notamment à cause de la dépendance de C^u, C^v et K à $\boldsymbol{u}_h$. Plusieurs stratégies de

linéarisation sont envisageables, dont les méthodes de points fixes et celle de Newton. C'est cette dernière que nous utiliserons et que nous présentons dans la section 1.2.4.

1.2.4 Linéarisation du système discret : méthode de Newton

Définissons tout d'abord les trois fonctionnelles suivantes :

$$\begin{cases} R_1(\boldsymbol{u},p,\boldsymbol{w}) &= \rho\big(\dfrac{\partial \boldsymbol{u}}{\partial t}+(\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u},\boldsymbol{w}\big)_{0,\Omega}+2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}-\big(p,\boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega} \\ & \quad -\langle \boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}-\rho\big(\boldsymbol{f},\boldsymbol{w}\big)_{0,\Omega}\,; \\ R_2(\boldsymbol{u},F,\varphi) &= \big(\dfrac{\partial F}{\partial t},\varphi\big)_{0,\Omega}+\big(\boldsymbol{u}\cdot\boldsymbol{\nabla}F,\varphi\big)_{0,\Omega}\,; \\ R_3(\boldsymbol{u},q) &= \big(\boldsymbol{\nabla}\cdot\boldsymbol{u},q\big)_{0,\Omega}. \end{cases}$$

Nous avons donc à résoudre le problème $R_i=0$, $i=1,2,3$. Sachant que $(\boldsymbol{u}^n,F^n,p^n)$ en est une solution approchée, on cherche dans la méthode de Newton les corrections $\delta\boldsymbol{u}$, δF et δp telles que :

$$\begin{cases} R_1(\boldsymbol{u}^n+\delta\boldsymbol{u},p^n+\delta p,\boldsymbol{w}) &= 0\,; \\ R_2(\boldsymbol{u}^n+\delta\boldsymbol{u},F^n+\delta F,\varphi) &= 0\,; \\ R_3(\boldsymbol{u}^n+\delta\boldsymbol{u},q) &= 0. \end{cases}$$

Cette réécriture du problème suggère de procéder à un développement de Taylor :

$$\begin{cases} 0 &= R_1(\boldsymbol{u}^n,p^n,\boldsymbol{w})+\dfrac{\partial R_1}{\partial \boldsymbol{u}}(\boldsymbol{u}^n,p^n,\boldsymbol{w})\delta\boldsymbol{u}+\dfrac{\partial R_1}{\partial p}(\boldsymbol{u}^n,p^n,\boldsymbol{w})\delta p+\cdots\,; \\ 0 &= R_2(\boldsymbol{u}^n,F^n,\varphi)+\dfrac{\partial R_2}{\partial \boldsymbol{u}}(\boldsymbol{u}^n,F^n,\varphi)\delta\boldsymbol{u}+\dfrac{\partial R_2}{\partial F}(\boldsymbol{u}^n,F^n,\varphi)\delta F+\cdots\,; \\ 0 &= R_3(\boldsymbol{u}^n,q)+\dfrac{\partial R_3}{\partial \boldsymbol{u}}(\boldsymbol{u}^n,q)\delta\boldsymbol{u}+\cdots. \end{cases} \quad (1.37)$$

Nous avons introduit ici la dérivée directionnelle d'une fonctionnelle, ou dérivée de Gâteaux :

$$\frac{\partial}{\partial \boldsymbol{x}}R(\boldsymbol{x})\delta\boldsymbol{x}=\frac{d}{d\xi}R(\boldsymbol{x}+\xi\delta\boldsymbol{x})\bigg|_{\xi=0}. \quad (1.38)$$

Ainsi,

$$\begin{aligned}
\frac{\partial R_1}{\partial \boldsymbol{u}}(\boldsymbol{u}^n, p^n, \boldsymbol{w})\delta\boldsymbol{u} &= \frac{d}{d\xi}\big[\big(\rho\frac{\partial(\boldsymbol{u}^n+\xi\delta\boldsymbol{u})}{\partial t} + \rho\big((\boldsymbol{u}^n+\xi\delta\boldsymbol{u})\cdot\boldsymbol{\nabla}\big)(\boldsymbol{u}^n+\xi\delta\boldsymbol{u}), \boldsymbol{w}\big)_{0,\Omega} + \\
&\quad 2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}^n+\xi\delta\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega} - \big(p^n, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega} - \\
&\quad \langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w}\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} - \rho\big(\boldsymbol{f}, \boldsymbol{w}\big)_{0,\Omega}\big]_{\xi=0} \\
&= \frac{d}{d\xi}\big[\rho\big(\frac{\partial \boldsymbol{u}^n}{\partial t}, \boldsymbol{w}\big)_{0,\Omega} + \xi\rho\big(\frac{\partial\delta\boldsymbol{u}}{\partial t}, \boldsymbol{w}\big)_{0,\Omega} + \rho\big((\boldsymbol{u}^n\cdot\boldsymbol{\nabla})\boldsymbol{u}^n, \boldsymbol{w}\big)_{0,\Omega} + \\
&\quad \xi^2\rho\big((\delta\boldsymbol{u}\cdot\boldsymbol{\nabla})\delta\boldsymbol{u}, \boldsymbol{w}\big)_{0,\Omega} + \xi\rho\big((\boldsymbol{u}^n\cdot\boldsymbol{\nabla})\delta\boldsymbol{u}, \boldsymbol{w}\big)_{0,\Omega} + \\
&\quad \xi\rho\big((\delta\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}^n, \boldsymbol{w}\big)_{0,\Omega} + 2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}^n), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega} + \\
&\quad 2\xi\mu\big(\dot{\boldsymbol{\gamma}}(\delta\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega} - \big(p^n, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega} + \\
&\quad \langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w}\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} - \rho\big(\boldsymbol{f}, \boldsymbol{w}\big)_{0,\Omega}\big]_{\xi=0} \\
&= \rho\big(\frac{\partial\delta\boldsymbol{u}}{\partial t}, \boldsymbol{w}\big)_{0,\Omega} + \rho\big((\boldsymbol{u}^n\cdot\boldsymbol{\nabla})\delta\boldsymbol{u}, \boldsymbol{w}\big)_{0,\Omega} + \rho\big((\delta\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}^n, \boldsymbol{w}\big)_{0,\Omega} + \\
&\quad 2\mu\big(\dot{\boldsymbol{\gamma}}(\delta\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}.
\end{aligned}$$

De façon similaire,

$$\begin{cases}
\dfrac{\partial R_1}{\partial p}(\boldsymbol{u}^n, p^n, \boldsymbol{w})\delta p = -\big(\delta p, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega}\,; \\
\dfrac{\partial R_2}{\partial \boldsymbol{u}}(\boldsymbol{u}^n, F^n, \varphi)\delta\boldsymbol{u} = \big(\delta\boldsymbol{u}\cdot\boldsymbol{\nabla}F^n, \varphi\big)_{0,\Omega}\,; \\
\dfrac{\partial R_2}{\partial F}(\boldsymbol{u}^n, F^n, \varphi)\delta F = \big(\dfrac{\partial\delta F}{\partial t}, \varphi\big)_{0,\Omega} + \big(\boldsymbol{u}^n\cdot\boldsymbol{\nabla}\delta F, \varphi\big)_{0,\Omega}\,; \\
\dfrac{\partial R_3}{\partial \boldsymbol{u}}(\boldsymbol{u}^n, q)\delta\boldsymbol{u} = \big(\boldsymbol{\nabla}\cdot\delta\boldsymbol{u}, q\big)_{0,\Omega}.
\end{cases}$$

En négligeant les termes d'ordre supérieur dans le système (1.37), on obtient ainsi la linéarisation de la forme faible des équations de modélisation :

$$\begin{cases} \rho\big(\frac{\partial \delta \boldsymbol{u}}{\partial t}, \boldsymbol{w}\big)_{0,\Omega} + \rho\big((\boldsymbol{u}^n \cdot \boldsymbol{\nabla})\delta \boldsymbol{u}, \boldsymbol{w}\big)_{0,\Omega} + \rho\big((\delta \boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}^n, \boldsymbol{w}\big)_{0,\Omega} + 2\mu\big(\dot{\boldsymbol{\gamma}}(\delta \boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega} \\ -\big(\delta p, \boldsymbol{\nabla} \cdot \boldsymbol{w}\big)_{0,\Omega} = -R_1(\boldsymbol{u}^n, p^n, \boldsymbol{w}) \, ; \\ \\ \big(\frac{\partial \delta F}{\partial t}, \varphi\big)_{0,\Omega} + \big(\delta \boldsymbol{u} \cdot \boldsymbol{\nabla} F^n, \varphi\big)_{0,\Omega} + \big(\boldsymbol{u}^n \cdot \boldsymbol{\nabla} \delta F, \varphi\big)_{0,\Omega} = -R_2(\boldsymbol{u}^n, F^n, \varphi) \, ; \\ \\ \big(\boldsymbol{\nabla} \cdot \delta \boldsymbol{u}, q\big)_{0,\Omega} = -R_3(\boldsymbol{u}^n, q). \end{cases}$$

En posant :

$$\begin{aligned} \delta u &\approx \sum_{j=1}^{N^u} \delta u_j \phi_j^u, & \delta v &\approx \sum_{j=1}^{N^v} \delta v_j \phi_j^v \, ; \\ \delta F &\approx \sum_{j=1}^{N^F} \delta F_j \phi_j^F, & \delta p &\approx \sum_{j=1}^{N^p} \delta p_j \phi_j^p, \end{aligned} \tag{1.39}$$

et en utilisant la même technique de discrétisation que celle utilisée dans la section précédente, nous obtenons le système matriciel global suivant :

$$\begin{pmatrix} T^u & 0 & 0 & 0 \\ 0 & T^v & 0 & 0 \\ 0 & 0 & T^F & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{\vec{\delta U}} \\ \dot{\vec{\delta V}} \\ \dot{\vec{\delta F}} \\ \dot{\vec{\delta P}} \end{pmatrix} +$$

$$\begin{pmatrix} C^{uu}(\boldsymbol{u}^n) + A^{uu} & C^{uv}(\boldsymbol{u}^n) + A^{uv} & 0 & (B^u)^T \\ C^{vu}(\boldsymbol{u}^n) + A^{vu} & C^{vv}(\boldsymbol{u}^n) + A^{vv} & 0 & (B^v)^T \\ K^u(\boldsymbol{u}^n) & K^v(\boldsymbol{u}^n) & K^F(\boldsymbol{u}^n) & 0 \\ B^u & B^v & 0 & 0 \end{pmatrix} \begin{pmatrix} \vec{\delta U} \\ \vec{\delta V} \\ \vec{\delta F} \\ \vec{\delta P} \end{pmatrix} = \begin{pmatrix} \vec{E}^u(\boldsymbol{u}^n) \\ \vec{E}^v(\boldsymbol{u}^n) \\ \vec{E}^F(\boldsymbol{u}^n) \\ \vec{E}^p(\boldsymbol{u}^n) \end{pmatrix}, \tag{1.40}$$

où, en notant $\boldsymbol{u}^n = (u^n \ v^n)^T$,

$$\begin{cases} C_{ij}^{uu}(\boldsymbol{u}^n) &= \rho\big(\boldsymbol{u}^n \cdot \boldsymbol{\nabla}\phi_j^u, \phi_i^u\big)_{0,\Omega} + \rho\big(\phi_j^u \dfrac{\partial u^n}{\partial x}, \phi_i^u\big)_{0,\Omega}\,; \\ C_{ij}^{uv}(\boldsymbol{u}^n) &= \rho\big(\phi_j^v \dfrac{\partial u^n}{\partial y}, \phi_i^u\big)_{0,\Omega}\,; \\ C_{ij}^{vu}(\boldsymbol{u}^n) &= \rho\big(\phi_j^u \dfrac{\partial v^n}{\partial x}, \phi_i^v\big)_{0,\Omega}\,; \\ C_{ij}^{vv}(\boldsymbol{u}^n) &= \rho\big(\boldsymbol{u}^n \cdot \boldsymbol{\nabla}\phi_j^v, \phi_i^v\big)_{0,\Omega} + \rho\big(\phi_j^v \dfrac{\partial v^n}{\partial y}, \phi_i^v\big)_{0,\Omega}, \end{cases}$$

$$\begin{cases} K_{ij}^u(\boldsymbol{u}^n) &= \big(\phi_j^u, \phi_i^F\big)_{0,\Omega} \dfrac{\partial F^n}{\partial x}\,; \\ K_{ij}^v(\boldsymbol{u}^n) &= \big(\phi_j^v, \phi_i^F\big)_{0,\Omega} \dfrac{\partial F^n}{\partial y}\,; \\ K_{ij}^F(\boldsymbol{u}^n) &= \big(\boldsymbol{u}^n \cdot \boldsymbol{\nabla}\phi_j^F, \phi_i^F\big)_{0,\Omega}, \end{cases}$$

$$\begin{cases} \vec{E}_i^u(\boldsymbol{u}^n) &= -\rho\big(\dfrac{\partial u^n}{\partial t}, \phi_i^u\big)_{0,\Omega} - \rho\big(\boldsymbol{u}^n \cdot \boldsymbol{\nabla}u^n, \phi_i^u\big)_{0,\Omega} - 2\mu\big(\dfrac{\partial u^n}{\partial x}, \dfrac{\partial \phi_i^u}{\partial x}\big)_{0,\Omega} + \\ & \quad \dfrac{1}{2}\big(\dfrac{\partial u^n}{\partial y} + \dfrac{\partial v^n}{\partial x}, \dfrac{\partial \phi_i^u}{\partial y}\big)_{0,\Omega} + \big(p^n, \phi_i^u\big)_{0,\Omega} + \langle t_x, \phi_i^u\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} + \rho\big(f_x, \phi_i^u\big)_{0,\Omega}\,; \\ \vec{E}_i^v(\boldsymbol{u}^n) &= -\rho\big(\dfrac{\partial u^n}{\partial t}, \phi_i^v\big)_{0,\Omega} - \rho\big(\boldsymbol{u}^n \cdot \boldsymbol{\nabla}v^n, \phi_i^v\big)_{0,\Omega} - 2\mu\big(\dfrac{\partial u^n}{\partial y}, \dfrac{\partial \phi_i^v}{\partial y}\big)_{0,\Omega} + \\ & \quad \dfrac{1}{2}\big(\dfrac{\partial u^n}{\partial y} + \dfrac{\partial v^n}{\partial x}, \dfrac{\partial \phi_i^v}{\partial x}\big)_{0,\Omega} + \big(p^n, \phi_i^v\big)_{0,\Omega} + \langle t_y, \phi_i^v\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} + \rho\big(f_y, \phi_i^v\big)_{0,\Omega}\,; \\ \vec{E}_i^F(\boldsymbol{u}^n) &= -\big(\dfrac{\partial F^n}{\partial t}, \phi_i^F\big)_{0,\Omega} - \big(\boldsymbol{u}^n \cdot \boldsymbol{\nabla}F^n, \phi_i^F\big)_{0,\Omega}\,; \\ \vec{E}_i^p(\boldsymbol{u}^n) &= -\big(\boldsymbol{\nabla} \cdot \boldsymbol{u}^n, \phi_i^p\big)_{0,\Omega}. \end{cases}$$

L'arrêt des itérations de Newton s'effectue lorsque :

- le rapport entre la norme de la correction et la norme de la solution est suffisamment petit :

$$\frac{\left\|\left(\delta\vec{U} \quad \delta\vec{V} \quad \delta\vec{P} \quad \delta\vec{F}\right)^T\right\|_2}{\left\|\left(\vec{\boldsymbol{U}}^{n+1} \quad \vec{\boldsymbol{V}}^{n+1} \quad \vec{\boldsymbol{P}}^{n+1} \quad \vec{\boldsymbol{F}}^{n+1}\right)^T\right\|_2} < \text{tol} ;$$

- le rapport entre la norme du résidu des équations et la norme du résidu initial est suffisamment petit :

$$\frac{\left\|\left(\vec{E}^u(\boldsymbol{u}^{n+1}) \quad \vec{E}^v(\boldsymbol{u}^{n+1}) \quad \vec{E}^F(\boldsymbol{u}^{n+1}) \quad \vec{E}^p(\boldsymbol{u}^{n+1})\right)^T\right\|_2}{\left\|\left(\vec{E}^u(\boldsymbol{u}^0) \quad \vec{E}^v(\boldsymbol{u}^0) \quad \vec{E}^F(\boldsymbol{u}^0) \quad \vec{E}^p(\boldsymbol{u}^0)\right)^T\right\|_2} < \text{tol}.$$

Le « suffisamment petit » se traduit par le choix d'une tolérance *tol* adaptée. Nous reviendrons sur le choix de la tolérance lors de la présentation des résultats numériques (cf. chapitre 3). Le tableau 1.3 résume les principales étapes qui ont abouti à ce système linéarisé. Par la suite, nous omettrons par soucis de simplicité la dépendance en $\boldsymbol{u}^n$ dans l'écriture des différentes matrices du système matriciel (1.40) et nous noterons donc simplement, par exemple, $C^{uu}(\boldsymbol{u}^n) = C^{uu}$.

1.2.4.1 Discrétisation temporelle : schéma de Gear

Pour pouvoir traiter numériquement des problèmes transitoires, nous devons également effectuer une discrétisation temporelle des termes transitoires. Comme nous l'avons vu dans la section précédente, l'équation de conservation de la quantité de mouvement s'écrit :

$$T\delta\dot{\vec{W}} + M\delta\vec{W} + N\delta\vec{P} = \vec{E}, \tag{1.41}$$

Tableau 1.3 – Linéarisation des équations discrétisées - Résumé

La forme faible linéarisée à l'itération n

$$
\begin{cases}
\rho\big(\dfrac{\partial \delta \boldsymbol{u}}{\partial t}, \boldsymbol{w}\big)_{0,\Omega} + \rho\big((\boldsymbol{u}^n \cdot \boldsymbol{\nabla})\delta \boldsymbol{u}, \boldsymbol{w}\big)_{0,\Omega} + \rho\big((\delta \boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}^n, \boldsymbol{w}\big)_{0,\Omega} + \\
2\mu\big(\dot{\boldsymbol{\gamma}}(\delta \boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega} - \big(\delta p, \boldsymbol{\nabla} \cdot \boldsymbol{w}\big)_{0,\Omega} = -R_1(\boldsymbol{u}^n, p^n, \boldsymbol{w}) ; \\
\big(\dfrac{\partial \delta F}{\partial t}, \varphi\big)_{0,\Omega} + \big(\delta \boldsymbol{u} \cdot \boldsymbol{\nabla} F^n, \varphi\big)_{0,\Omega} + \big(\boldsymbol{u}^n \cdot \boldsymbol{\nabla} \delta F, \varphi\big)_{0,\Omega} = \\
-R_2(\boldsymbol{u}^n, F^n, \varphi) ; \\
\big(\boldsymbol{\nabla} \cdot \delta \boldsymbol{u}, q\big)_{0,\Omega} = -R_3(\boldsymbol{u}^n, p^n, q).
\end{cases}
$$

La linéarisation des équations discrétisées

$$
\begin{pmatrix} T^u & 0 & 0 & 0 \\ 0 & T^v & 0 & 0 \\ 0 & 0 & T^F & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{\vec{U}} \\ \dot{\vec{V}} \\ \dot{\vec{F}} \\ \dot{\vec{P}} \end{pmatrix} +
$$

$$
\begin{pmatrix} C^{uu} + A^{uu} & C^{uv} + A^{uv} & 0 & (B^u)^T \\ C^{vu} + A^{vu} & C^{vv} + A^{vv} & 0 & (B^v)^T \\ K^u & K^v & K^F & 0 \\ B^u & B^v & 0 & 0 \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{V} \\ \vec{F} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{E}^u \\ \vec{E}^v \\ \vec{E}^F \\ \vec{E}^p \end{pmatrix}.
$$

où

$$T = \begin{pmatrix} T^u & 0 \\ 0 & T^v \end{pmatrix}, \quad M = \begin{pmatrix} C^{uu} + A^{uu} & C^{uv} + A^{uv} \\ C^{vu} + A^{vu} & C^{vv} + A^{vv} \end{pmatrix}, \quad N = \begin{pmatrix} (B^u)^T \\ (B^v)^T \end{pmatrix}$$

$$\delta\vec{W} = \begin{pmatrix} \delta\vec{U} \\ \delta\vec{V} \end{pmatrix}, \quad \delta\dot{\vec{W}} = \begin{pmatrix} \delta\dot{\vec{U}} \\ \delta\dot{\vec{V}} \end{pmatrix} \text{ et } \vec{E} = \begin{pmatrix} \vec{E}^u \\ \vec{E}^v \end{pmatrix}.$$

La discrétisation en temps peut alors être effectuée selon plusieurs schémas. Nous présentons ici le schéma de Gear (Gresho et al. 1999) qui est implicite, d'ordre 2 et A-stable. Cette approche semi-discrète peut être utilisée avec d'autres schémas d'intégration en temps. Supposons que le système (1.41) a été résolu aux deux temps $t = t_{k-1}$ et $t = t_k$. Les solutions sont alors notées respectivement $\delta\vec{W}^{k-1}$ et $\delta\vec{W}^{k}$. Le schéma de Gear peut alors être exprimé comme :

$$\delta\dot{\vec{W}}^{k+1} \approx \frac{3\delta\vec{W}^{k+1} - 4\delta\vec{W}^{k} + \delta\vec{W}^{k-1}}{2\Delta t}, \tag{1.42}$$

où $\Delta t = t_k - t_{k-1}$. Ainsi, si nous reprenons le système (1.41) en indiçant les inconnues par $k+1$ pour spécifier que nous recherchons la solution au temps t_{k+1}, nous avons :

$$T\delta\dot{\vec{W}}^{k+1} + M\delta\vec{W}^{k+1} + N\delta\vec{P}^{k+1} = \vec{E}$$

$$\Leftrightarrow$$

$$T\left(\frac{3\delta\vec{W}^{k+1} - 4\delta\vec{W}^{k} + \delta\vec{W}^{k-1}}{2\Delta t}\right) + M\delta\vec{W}^{k+1} + N\delta\vec{P}^{k+1} = \vec{E}$$

$$\Leftrightarrow$$

$$(3T + 2\Delta t M)\delta\vec{W}^{k+1} + 2\Delta t N\delta\vec{P}^{k+1} = 2\Delta t\vec{E} + 4T\delta\vec{W}^{k} - T\delta\vec{W}^{k-1}.$$

Ainsi, pour chaque pas de temps t_{k+1}, le système (1.41) se réécrit :

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta\vec{U}^{k+1} \\ \delta\vec{P}^{k+1} \end{pmatrix} = \begin{pmatrix} \vec{E}^k \\ \vec{Q} \end{pmatrix}, \tag{1.43}$$

où

$$A = \begin{pmatrix} 3T^u + 2\Delta t(C^{uu} + A^{uu}) & 2\Delta t(C^{uv} + A^{uv}) & 0 \\ 2\Delta t(C^{vu} + A^{vu}) & 3T^v + 2\Delta t(C^{vv} + A^{vv}) & 0 \\ K^u & K^v & K^F 2\Delta t B^u \end{pmatrix},$$

$$B = \begin{pmatrix} 2\Delta t B^u & 2\Delta t B^v \end{pmatrix}, \qquad \vec{E}^k = \begin{pmatrix} 2\Delta t\vec{E}^u + 4T\delta\vec{U}^k - T\delta\vec{U}^{k-1} \\ 2\Delta t\vec{E}^v + 4T\delta\vec{V}^k - T\delta\vec{V}^{k-1} \\ \vec{E}^F \end{pmatrix},$$

$$\vec{Q} = 2\Delta t\vec{E}^p \qquad \text{et} \qquad \delta\vec{\boldsymbol{U}}^{k+1} = \begin{pmatrix} \delta\vec{U}^{k+1} \\ \delta\vec{V}^{k+1} \\ \delta\vec{P}^{k+1} \end{pmatrix}.$$

Le tableau 1.4 résume les principales étapes qui nous ont conduites au système (1.43). Ce système est linéaire et plusieurs stratégies de résolution sont dès lors envisageables. La prochaine section va nous permettre de présenter la formulation dite *mixte* et la résolution par la méthode d'*Uzawa*.

1.2.5 Stratégies de résolution

Les sections précédentes nous ont permis de transformer notre problème continu de modélisation d'écoulement à surface libre en un problème discret et linéaire, et plus précisément en un système matriciel à résoudre. Dès lors, il convient de s'interroger sur des méthodes de résolution efficaces de ces systèmes linéaires. Les deux prochaines sous-sections traitent de deux types de formulation : la formulation *mixte* et la formulation d'*Uzawa*.

1.2.5.1 Formulation mixte

La formulation mixte consiste à trouver, à chaque itération de la méthode de Newton, le vecteur de correction $(\delta\vec{\boldsymbol{U}}, \delta\vec{P})$ en résolvant de façon couplée le système

Tableau 1.4 – Application d'un schéma de discrétisation temporelle - Résumé

La linéarisation des équations discrétisées

$$\begin{pmatrix} T^u & 0 & 0 & 0 \\ 0 & T^v & 0 & 0 \\ 0 & 0 & T^F & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{\vec{U}} \\ \dot{\vec{V}} \\ \dot{\vec{F}} \\ \dot{\vec{P}} \end{pmatrix} +$$

$$\begin{pmatrix} C^{uu} + A^{uu} & C^{uv} + A^{uv} & 0 & (B^u)^T \\ C^{vu} + A^{vu} & C^{vv} + A^{vv} & 0 & (B^v)^T \\ K^u & K^v & K^F & 0 \\ B^u & B^v & 0 & 0 \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{V} \\ \vec{F} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{E}^u \\ \vec{E}^v \\ \vec{E}^F \\ \vec{E}^p \end{pmatrix}.$$

L'application d'un schéma de discrétisation temporelle

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta\vec{\boldsymbol{U}}^{k+1} \\ \delta\vec{P}^{k+1} \end{pmatrix} = \begin{pmatrix} \vec{E}^k \\ \vec{Q} \end{pmatrix}.$$

linéaire (1.43) par une méthode directe ou itérative. Ne considérons, pour simplifier les choses, que le problème de Stokes (on enlève les matrices de convection C) stationnaire (on enlève les termes transitoires T) dans le cas d'un seul fluide (on enlève les matrices de pseudo-concentration K). Nous pouvons alors résoudre le problème en variables primitives plutôt qu'en corrections, ce qui nous donne le système :

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{E} \\ \vec{0} \end{pmatrix}, \tag{1.44}$$

où A est une matrice symétrique définie positive :

$$\begin{cases} A^T = A\,; \\ \boldsymbol{x}^T A \boldsymbol{x} > 0,\ \forall \boldsymbol{x} \in \mathbb{R}^n. \end{cases}$$

Nous avons dans le système (1.44) une contrainte homogène $B\vec{U} = \vec{0}$. Si on considère le cas général où la contrainte est cette fois-ci non homogène, ce système s'écrit :

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{E} \\ \vec{Q} \end{pmatrix}. \tag{1.45}$$

Du point de vue de l'optimisation, notre système matriciel peut être vu comme résultant d'un problème de minimisation sous contraintes. En effet, définissons le problème d'optimisation suivant :

$$\begin{cases} \min\limits_{\vec{U}} & \frac{1}{2}\vec{U}^T A\vec{U} - \vec{U}^T\vec{E}\,; \\ \text{s.c.} & B\vec{U} = \vec{Q}. \end{cases} \tag{1.46}$$

Le problème quadratique (1.46) est un problème de minimisation avec contrainte d'égalité. Définissons le lagrangien $L(\vec{U}, \vec{P})$ associé à ce problème, où $\vec{P}$ sont les composantes des multiplicateurs de Lagrange :

$$L(\vec{U}, \vec{P}) = \frac{1}{2}\vec{U}^T A\vec{U} - \vec{U}^T\vec{E} + \vec{P}^T(B\vec{U} - \vec{Q}).$$

Les conditions d'optimalité du premier ordre s'obtiennent en imposant la nullité du gradient du lagrangien L :

$$\nabla L(\vec{U},\vec{P})=\vec{0} \Leftrightarrow \begin{cases} \dfrac{\partial L}{\partial \vec{U}}=0\,; \\ \dfrac{\partial L}{\partial \vec{P}}=0 \end{cases}$$

$$\Leftrightarrow \begin{cases} A\vec{U}-\vec{E}+B^T\vec{P}=0\,; \\ B\vec{U}-\vec{Q}=0 \end{cases}$$

$$\Leftrightarrow \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} \vec{U} \\ \vec{P} \end{pmatrix}=\begin{pmatrix} \vec{E} \\ \vec{Q} \end{pmatrix}.$$

La matrice A étant définie positive sur $ker(B)$, les conditions (1.45) sont également suffisantes pour avoir l'optimalité dans (1.46). Ce problème de minimisation est donc équivalent au système linéaire (1.45) (Fortin and Glowinksy 1983). La méthode mixte peut dans ce cas être vue comme le problème quadratique (1.46) dans lequel la pression joue le rôle de multiplicateur de Lagrange.

1.2.5.2 Méthode d'Uzawa

Pour présenter la méthode d'Uzawa, revenons au problème d'optimisation (1.46). Ce dernier peut être également transformé en problème de point-selle du Lagrangien :

$$\min_{\vec{U}}\max_{\vec{P}}\frac{1}{2}\vec{U}^TA\vec{U}-\vec{U}^T\vec{E}+(\vec{U}^TB^T-\vec{Q}^T)\vec{P}. \tag{1.47}$$

Pour $\vec{P}$ fixé, le minimum est atteint en $\vec{U}_{\vec{P}}$ solution de $A\vec{U}_{\vec{P}}=\vec{E}-B^T\vec{P}$. Ainsi,

$$\vec{U}_{\vec{P}}=A^{-1}(\vec{E}-B^T\vec{P}) \Leftrightarrow \vec{U}_{\vec{P}}^T=\vec{E}^TA^{-1}-\vec{P}^TBA^{-1}. \tag{1.48}$$

En remplaçant (1.48) dans (1.47), nous obtenons :

$$\begin{aligned}
& \max_{\vec{P}} \frac{1}{2}\vec{U}_{\vec{P}}^T A\vec{U}_{\vec{P}} - \vec{U}_{\vec{P}}^T(\vec{E} - B^T\vec{P}) - \vec{Q}^T\vec{P} \\
\Leftrightarrow \quad & \max_{\vec{P}} \frac{1}{2}(\vec{E}^T A^{-1} - \vec{P}^T BA^{-1})(\vec{E} - B^T\vec{P}) - (\vec{E}^T A^{-1} - \vec{P}^T BA^{-1})(\vec{E} - B^T\vec{P}) - \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vec{Q}^T\vec{P} \\
\Leftrightarrow \quad & \max_{\vec{P}} -\frac{1}{2}(\vec{E}^T A^{-1} - \vec{P}^T BA^{-1})(\vec{E} - B^T\vec{P}) - \vec{Q}^T\vec{P} \\
\Leftrightarrow \quad & -\min_{\vec{P}} \frac{1}{2}(\vec{E}^T A^{-1} - \vec{P}^T BA^{-1})(\vec{E} - B^T\vec{P}) - \vec{Q}^T\vec{P} \\
\Leftrightarrow \quad & -\min_{\vec{P}} \frac{1}{2}(\vec{E}^T AE - \vec{E}^T A^{-1}B^T\vec{P} - \vec{P}^T BA^{-1}\vec{E} + \vec{P}^T BA^{-1}B^T\vec{P}) - \vec{P}^T\vec{Q} \quad (1.49) \\
\Leftrightarrow \quad & -\min_{\vec{P}} \frac{1}{2}(\vec{P}^T(BA^{-1}B^T)\vec{P}) - \vec{P}^T(BA^{-1}\vec{E} - \vec{Q}). \quad (1.50)
\end{aligned}$$

En passant de (1.49) à (1.50), nous avons enlevé le terme $\vec{E}^T A^{-1}\vec{E}$ qui ne dépend pas de $\vec{P}$ et n'influe donc pas sur le résultat de la minimisation. Si B est de rang maximal, $BA^{-1}B^T$ est définie positive. Ainsi, le minimum du problème (1.50) est atteint lorsque

$$BA^{-1}B^T\vec{P} = BA^{-1}\vec{E} - \vec{Q}.$$

On peut résoudre le système matriciel ci-dessus en appliquant par exemple une méthode de plus forte pente à pas fixe. On en déduit l'algorithme d'Uzawa 1.1 (Fortin and Glowinksy 1983, Arrow et al. 1958). Le calcul de A^{-1} n'est évidemment pas envisageable. Quelques modifications s'imposent donc. Remarquons tout d'abord que

$$\vec{R} = \vec{Q} - BA^{-1}\vec{E} + BA^{-1}B^T\vec{P} = \vec{Q} - B(A^{-1}(\vec{E} - B^T\vec{P})) = \vec{Q} - B\vec{U}.$$

Il s'agit du résidu de la contrainte pour la solution intermédiaire $\vec{U}$. Pour le calcul du paramètre de descente α, le vecteur $\vec{W} = A^{-1}B^T\vec{R}$ peut être introduit. On obtient ainsi une deuxième version de l'algorithme d'Uzawa (Algorithme 1.2). Un moyen efficace d'augmenter la vitesse de convergence de la méthode d'Uzawa est d'employer

Algorithme d'Uzawa - Version 1

$\vec{P} = \vec{P}_0$.

Pour $k = 0, 1, \ldots$

$$\vec{R} = -BA^{-1}\vec{E} + \vec{Q} + BA^{-1}B^T\vec{P},$$

$$\alpha = \frac{\vec{R}^T\vec{R}}{\vec{R}^T(BA^{-1}B^T)\vec{R}},$$

$$\vec{P} = \vec{P} - \alpha\vec{R},$$

Résoudre $A\vec{U} = \vec{E} - B^T\vec{P}$.

Algorithme 1.1 – Uzawa - Version 1.

Algorithme d'Uzawa - Version 2

$\vec{P} = \vec{P}_0$.

Pour $k = 0, 1, \ldots$

Résoudre $A\tilde{\vec{U}} = \vec{E} - B^T\vec{P}$,

$$\vec{R} = \vec{Q} - B\tilde{\vec{U}},$$

Calculer $\vec{Z} = B^T\vec{R}$,

Résoudre $A\vec{W} = \vec{Z}$,

$$\alpha = \frac{\vec{R}^T\vec{R}}{\vec{Z}^T\vec{W}},$$

$$\vec{P} = \vec{P} - \alpha\vec{R}.$$

$\vec{U} = \tilde{\vec{U}}$.

Algorithme 1.2 – Uzawa - Version 2.

une technique de Lagrangien augmenté. En effet, on peut montrer que la solution du système (1.45) est également solution du système

$$\begin{pmatrix} A_r & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{E}_r \\ \vec{Q} \end{pmatrix},$$

où $A_r = A + rB^TB$ et $E_r = E + rB^TD$ (Fortin and Glowinksy 1983).
Il suffit donc de reprendre l'algorithme en remplaçant A par A_r et E par E_r. De plus, lorsque le coefficient de pénalisation r est suffisamment élevé, l'algorithme peut grandement se simplifier car $\alpha \approx r$. En effet, lorsque r est grand, on peut écrire $A_r = A + rB^TB \approx rB^TB$, ce qui entraîne :

$$\begin{aligned} & A_r\vec{W} = B^T\vec{R} \\ \Leftrightarrow \quad & (A + rB^TB)\vec{W} = B^T\vec{R} \\ \Leftrightarrow \quad & rB^TB\vec{W} \approx B^T\vec{R} \\ \Leftrightarrow \quad & B^T(rB\vec{W} - \vec{R}) \approx \vec{0}. \end{aligned}$$

Si l'on suppose que B est de rang maximal, on obtient : $rB\vec{W} = \vec{R} \Leftrightarrow B\vec{W} = \dfrac{\vec{R}}{r}$.
Ainsi, $\alpha = \dfrac{\vec{R}^T\vec{R}}{\vec{R}^T(B\vec{W})} \approx \dfrac{\vec{R}^T\vec{R}}{\vec{R}^T(\vec{R}/r)} = r\dfrac{\vec{R}^T\vec{R}}{\vec{R}^T\vec{R}} = r.$
Si on réécrit l'algorithme 1.2 en prenant en compte ces simplifications, nous arrivons à l'algorithme 1.3. En terme de corrections et en ajoutant des critères d'arrêts adéquats, on obtient l'algorithme 1.4. Lorsque la matrice A est non symétrique, le même algorithme peut être dérivé en se basant sur la méthode du résidu minimal et en effectuant les mêmes simplifications pour r élevé (Fortin and Glowinksy 1983).

Algorithme d'Uzawa - Version 3

$\vec{P} = \vec{P}_0$.

Pour $k = 0, 1, \ldots$

Résoudre $A_r \tilde{\vec{U}} = \vec{E}_r - B^T \vec{P}$,

$\vec{R} = \vec{Q} - B\tilde{\vec{U}}$,

$\vec{P} = \vec{P} - r\vec{R}$.

$\vec{U} = \tilde{\vec{U}}$.

Algorithme 1.3 – Uzawa - Version 3.

Algorithme d'Uzawa - Version corrections

Approximations initiales $\vec{P} = \vec{P}_0$, $\vec{U} = \vec{U}_0$,

Résidu (mouvement) : $\vec{R}_1 = A\vec{U}_0 + B^T\vec{P}_0 - \vec{E}$,

Résidu (masse) : $\vec{R}_2 = B\vec{U}_0 - \vec{Q}$.

Faire

Résoudre $(A + rB^TB)\delta\vec{U} = -\vec{R}_1 - rB^T\vec{R}_2$,

Mise-à-jour de la vitesse et de la pseudo-concentration : $\vec{U} = \vec{U} + \delta\vec{U}$,

Mise-à-jour du résidu (masse) : $\vec{R}_2 = B\vec{U} - \vec{Q}$,

Mise-à-jour du résidu (mouvement) : $\vec{R}_1 = A\vec{U} + B^T\vec{P} - \vec{E}$,

Tant que ($\| \vec{R}_1 \| > \epsilon$) et ($\| \delta\vec{U} \| > \epsilon$) et ($\| \vec{R}_2 \| > \epsilon$).

Algorithme 1.4 – Uzawa - Version 4.

CHAPITRE 2

MÉTHODES DIRECTES ET ITÉRATIVES POUR LA RÉSOLUTION DE SYSTÈMES LINÉAIRES DE GRANDE TAILLE

Le premier chapitre de ce mémoire nous a permis de présenter les équations régissant les écoulements multifluides et incompressibles. Nous avons vu que la méthode des éléments finis appliquée à ces équations mène à un système linéaire à résoudre à chaque itération de Newton, et ce quelle que soit la formulation utilisée. Dans le cas d'une formulation mixte, les matrices sont de la forme

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix},$$

et dans le cas de la méthode d'Uzawa, on a

$$A + rB^T B,$$

où r est un paramètre de pénalisation ($\sim 10^6$) et où A et B sont des matrices de grande taille.

Deux grandes familles de méthodes sont envisageables pour résoudre numériquement des systèmes d'équations algébriques linéaires : les méthodes *directes* et les méthodes *itératives*. Chaque approche possède ses avantages et ses inconvénients. Les premières permettent de trouver la solution en un nombre d'opérations connu à l'avance et avec une erreur relative de l'ordre de la précision machine dans la plupart des cas. En revanche, l'utilisation de l'espace mémoire que nécessite le processus de factorisation peut devenir très important pour des matrices de grande taille, même si ces dernières sont creuses (*i.e.* avec une proportion importante d'éléments nuls). Les secondes ont

l'avantage de ne nécessiter que des produits matrice-vecteur. Ceci est d'autant plus avantageux lorsque l'on ne stocke que les éléments non nuls. En revanche, la convergence peut être lente et dans certains cas, elle n'est même pas assurée (nous reviendrons sur ce point de détail un peu plus loin dans ce chapitre lorsque nous parlerons des « breakdown »). De plus, l'usage d'un préconditionneur adéquat, *i.e.* une matrice qui permet d'améliorer le conditionnement d'un système linéaire et d'accélérer la convergence des méthodes itératives, s'avère souvent essentiel. Souvent, celui-ci est dicté par l'application. Ces deux approches sont traitées plus en détail dans les sections qui suivent.

2.1 Méthodes directes

Nous nous intéressons ici à la résolution du système $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$, où $\mathcal{A} = (a_{ij})$ est une matrice de taille $N \times N$, à l'aide d'une factorisation LU (*Lower/Upper triangular*) (Trefethen and Bau 1997). Cette dernière génère deux matrices dont l'une est triangulaire inférieure et l'autre est triangulaire supérieure. Quatre étapes sont nécessaires pour la résolution du système :

- Réduction du profil et de la largeur de bande de la matrice ;
- Stockage de la matrice ;
- Factorisation LU ;
- Résolution du système via les facteurs.

La factorisation de la matrice $\mathcal{A}$ peut être de la forme :

$$P_F\mathcal{A} = LU \quad \text{ou} \quad P_F\mathcal{A}Q_F = LU,$$

où P_F est une matrice de permutation et Q_F est une matrice de renumérotation. Notons qu'il est fréquent d'utiliser un vecteur $\boldsymbol{p}_F$ plutôt que la matrice P_F pour stocker les permutations, de sorte que $(P_F\mathcal{A})_{ij} = \mathcal{A}(\boldsymbol{p}_F(i), j)$, $i = 1, 2, \ldots, N$. La décomposition LU n'est pas unique. Deux algorithmes sont couramment utilisés :

l'algorithme de *Crout* qui fournit une matrice U dont les éléments diagonaux sont égaux à 1 et l'algorithme de *Doolittle* qui fournit une matrice L dont les éléments diagonaux sont égaux à 1.

2.1.1 Réduction du profil et de la largeur de bande

Le profil et la largeur de bande d'une matrice $\mathcal{A}$ sont deux mesures de la « distance » à la diagonale. Plus ces quantités sont petites, moindres sont les besoins en mémoire pour le stockage. Dans la méthode des éléments finis, le profil et la largeur de bande sont directement liés à la façon dont sont numérotés les degrés de liberté. Nous allons donc appliquer des algorithmes de renumérotation des degrés de liberté avant même que la matrice soit assemblée.
Notons respectivement β et δ la largeur de bande et de profil de la matrice $\mathcal{A}$. Ces deux quantités sont définies comme suit :

$$\beta = \max_{a_{ij} \neq 0} |i - j| \qquad \text{et} \qquad \delta = \sum_{i=1}^{n} \delta_i,$$

où $\delta_i = i - \min\{j \mid a_{ij} \neq 0\}$. Pour chaque élément a_{ij} non nul de la matrice $\mathcal{A}$, on mesure la distance qui le sépare de l'élément diagonal (*i.e.* le nombre d'éléments entre a_{ij} inclus, et a_{ii} non inclus). La largeur de bande représentera ainsi le maximum de l'ensemble des distances, alors que le profil représentera la somme des distances liées au premier élément non nul de chaque ligne.

Exemple : Soit la matrice 6×6 suivante (les éléments non nuls sont représentés

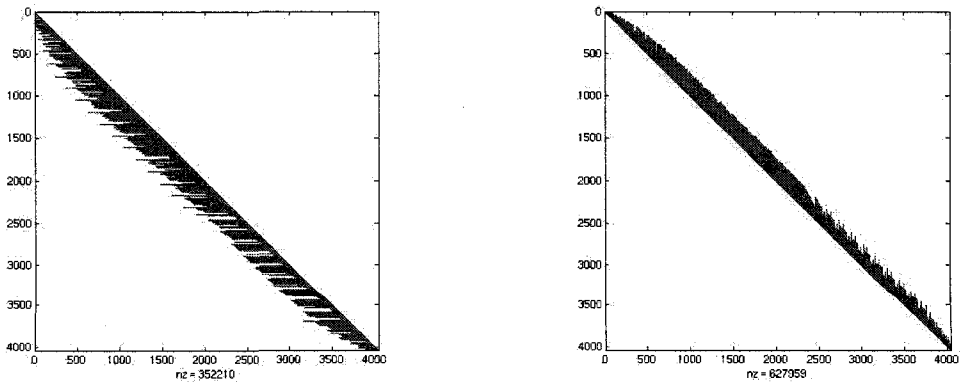
symboliquement par des $\times$) :

$$\mathcal{A} = \begin{pmatrix} \times & \times & \times & & \times & \\ & \times & \times & \times & & \\ & \times & \times & \times & \times & \\ & \times & \times & \times & \times & \\ & & \times & \times & \times & \\ & & \times & & \times & \times \end{pmatrix}.$$

On a alors :

$$\begin{aligned} \beta &= |1-5| = 4\,; \\ \delta &= 0+0+1+2+2+3 = 8. \end{aligned}$$

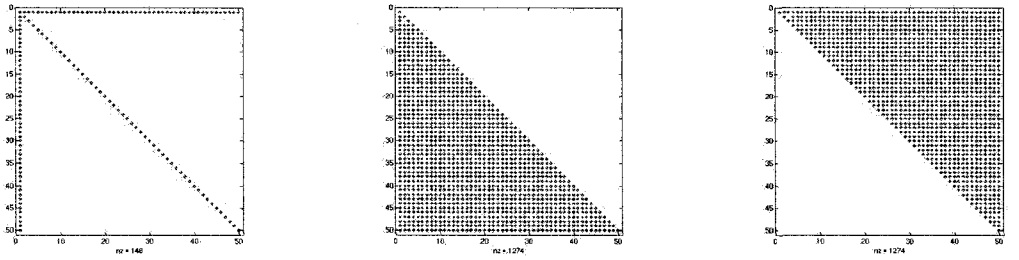
Les matrices à faible profil ou à faible largeur de bande sont stockées plus efficacement, par l'utilisation par exemple du format « skyline » qui fait l'objet de la prochaine section. De plus, tel qu'illustré à la figure 2.1, le processus de décomposition LU de ces matrices préserve dans la plupart des cas cette structure dans les facteurs résultants, ce qui permet une résolution plus rapide lors des remontées et des descentes triangulaires.



(a) Facteur L (b) Facteur U

Figure 2.1 – Décomposition LU d'une matrice bande.

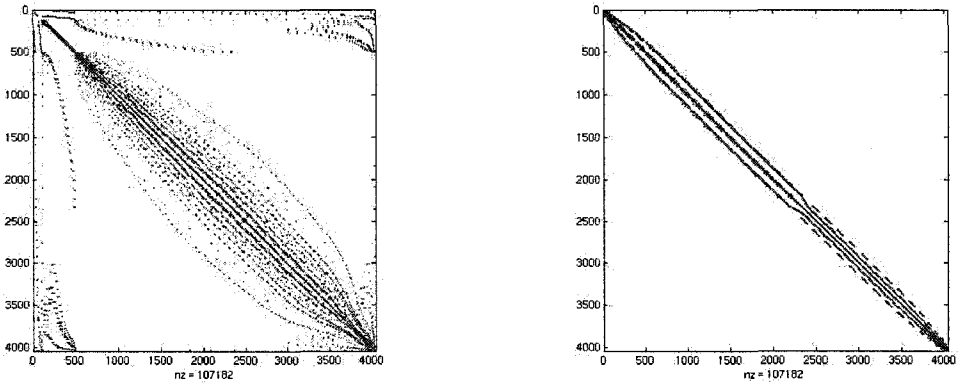
A contrario, des matrices creuses à profil ou à largeur de bande élevés entraînent souvent des facteurs denses (cf. Figure 2.2).



(a) Matrice flèche (b) Facteur L (c) Facteur U

Figure 2.2 – Décomposition LU d'une matrice flèche : $N = 50$, $\beta = 50$, $\delta = 1225$.

C'est le phénomène de « *fill-in* » . Plusieurs algorithmes proposent des stratégies de diminution du « *fill-in* » (tel qu'illustré à la figure 2.3) en utilisant des algorithmes de renumérotation (Gibbs et al. 1974).



(a) Matrice non renumérotée (b) Matrice renumérotée

Figure 2.3 – Renumérotation par l'algorithme de Gibbs-Poole-Stockmeyer.

2.1.2 Stockage

Plusieurs formats de stockage permettent de réduire considérablement les besoins en espace mémoire lorsque les matrices sont creuses. Citons parmis les plus utilisés :

- le format coordonnées : on stocke dans trois tableaux `AA`, `JR` et `JC` respectivement les valeurs non nulles, les indices de colonnes et les indices de lignes correspondants ;
- le format en lignes compressées CSR (***C**ompressed **S**parse **R**ow*) : on stocke dans deux tableaux `AA` et `JA` respectivement les valeurs non nulles et les indices de colonnes correspondants. Un troisième tableau `IA` contient dans la case i un pointeur vers le début de la ligne i dans les tableaux `AA` et `JA` ;
- le format « skyline » (traduit littéralement par format en *ligne de ciel*).

Ce dernier format de stockage est bien adapté aux matrices creuses à faible profil ou à faible largeur de bande. Ce format consiste à utiliser deux tableaux pour représenter une matrice. Le premier, appelons-le pour les illustrations qui suivent `val`, stocke les valeurs de la matrice dans un ordre bien particulier. En effet, on parcourt chaque ligne i de la matrice et on stocke dans l'ordre les ensembles suivants :

- $S_{i,1} = \{a_{ij} \mid k_i \leq j < i\}$;
- $S_{i,2} = \{a_{ji} \mid k_i \leq j < i\}$;
- $S_{i,3} = a_{ii}$,

où $k_i = \min\{k \leq i \mid a_{ik} \neq 0 \text{ ou } a_{ki} \neq 0\}$.

Le deuxième tableau, `diag`, stocke pour chaque ligne de la matrice, la position de l'élément diagonal dans le tableau `val`.

Exemple 1 :
Soit la matrice

$$\begin{pmatrix} 1 & 0 & 3 \\ 0 & 5 & 0 \\ 0 & 0 & 2 \end{pmatrix}.$$

Nous avons alors :

- $i = 1 \Rightarrow k_1 = 1 \Rightarrow S_{1,1} = S_{1,2} = \emptyset$ et $S_{1,3} = \{1\}$;
- $i = 2 \Rightarrow k_2 = 2 \Rightarrow S_{2,1} = S_{2,2} = \emptyset$ et $S_{2,3} = \{5\}$;
- $i = 3 \Rightarrow k_3 = 1 \Rightarrow S_{3,1} = \{0, 0\}$, $S_{3,2} = \{3, 0\}$ et $S_{3,3} = \{2\}$.

Les structure de données correspondantes sont alors :

`val` :	1	5	0	0	3	0	2
`diag` :	1	2	7				

Exemple 2 :

Soient les structures de données suivantes :

`val` :	7	4	3	9	6	8	1	2	5
`diag` :	1	4	9						

Le tableau `diag` nous indique que la matrice est de taille 3 et que les éléments diagonaux sont respectivement en position 1, 4 et 9 dans le tableau `val`. Ainsi :

- $S_{1,3} = \{7\} \Rightarrow S_{1,1} = S_{1,2} = \emptyset$;
- $S_{2,3} = \{9\} \Rightarrow S_{2,1} = \{4\}$ et $S_{2,2} = \{3\}$;
- $S_{3,3} = \{5\} \Rightarrow S_{3,1} = \{6, 8\}$ et $S_{3,2} = \{1, 2\}$.

La matrice correspondante est donc :

$$\begin{pmatrix} 7 & 3 & 1 \\ 4 & 9 & 2 \\ 6 & 8 & 5 \end{pmatrix}.$$

Le premier exemple illustre le principal inconvénient de ce format de stockage. En effet, les valeurs a_{31} et a_{32} de la matrice sont nulles mais sont tout de même stockées

dans la structure de données `val`. Cela est dû au fait que l'élément a_{13} est non nul. Ainsi, comme nous le verrons dans le chapitre 3, de nombreux 0 peuvent être stockés pour des matrices de grande taille. Le format « skyline » reste tout de même particulièrement adapté aux matrices à faible profil ou à faible largeur de bande non seulement parce que les besoins en mémoire pour le stockage de la matrice sont moindres, mais aussi car les algorithmes de résolution sont plus efficaces (moins d'opérations). C'est ce dernier aspect qui fait l'objet de la section suivante.

2.1.3 Factorisation et résolution

La résolution d'un système matriciel par une méthode directe passe par un algorithme de factorisation efficace. Nous utilisons ici l'algorithme de Doolittle 2.1 (Ralston and Rabinowitz 1978) qui effectue la décomposition LU avec permutation de la matrice $\mathcal{A} = (a_{ij})$ de taille $N \times N$ en notation compacte, *i.e.* les données des facteurs sont écrites dans les mêmes structures de données utilisées pour stocker la matrice $\mathcal{A}$. Le triangle inférieur de $\mathcal{A}$, sans les éléments diagonaux, contient la matrice L et le triangle supérieur, avec les éléments diagonaux, contient la matrice U. Nous notons $\mathcal{A}(i : N, j)$ le vecteur colonne dont les éléments sont les $\mathcal{A}(k, j)$, $k = i, \ldots, N$. On peut optimiser cet algorithme en prenant en compte les avantages du format « skyline » :

1. L'algorithme de Doolitlle effectue la factorisation en calculant le $i^{\text{ème}}$ pivot, puis la $i^{\text{ème}}$ ligne de U et enfin la $i^{\text{ème}}$ colonne de L. On peut visualiser le déroulement de cette procédure sur la figure 2.4.

Algorithme de Doolittle en notation compacte

Pour $j = 1, 2, \ldots, N$

Pour $i = 1, \ldots, j-1$

$\mathcal{A}(i,j) = \mathcal{A}(p_F(i), j)$,

$\mathcal{A}(i,j) = \mathcal{A}(i,j)/\mathcal{A}(i,i)$,

$\mathcal{A}(i+1:N, j) = \mathcal{A}(i+1:N, j) - \mathcal{A}(i+1:N, i) * A(i,j)$,

Trouver p, $j \leq p \leq N$, tel que $|\mathcal{A}(p,j)| = \| \mathcal{A}(j:N, j) \|_\infty$,

$\mathcal{A}(j,j) = \mathcal{A}(p,j)$,

$\boldsymbol{p}_F(j) = p$.

Algorithme 2.1 – Doolittle.

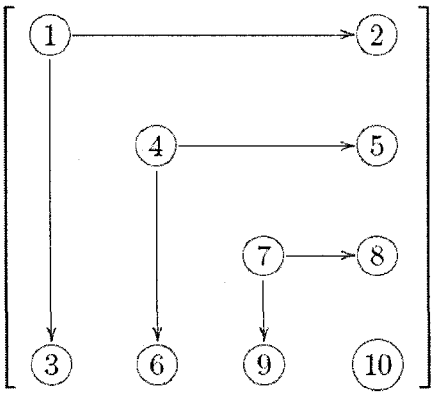


Figure 2.4 – Ordre des opérations dans l'algorithme de Doolittle classique.

Un ordre plus en adéquation avec le stockage en ligne de ciel serait de calculer la $i^{\text{ème}}$ ligne de L, puis la $i^{\text{ème}}$ colonne de U et enfin le $i^{\text{ème}}$ pivot, comme illustré par la figure 2.5, ce qui suggère une première amélioration de l'algorithme.

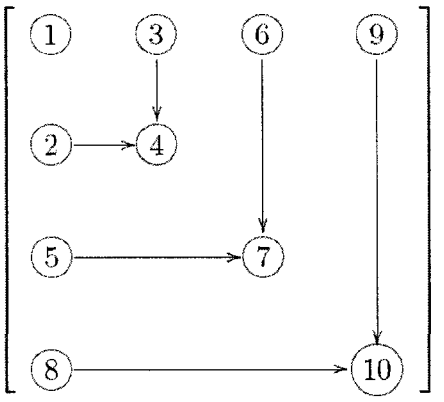


Figure 2.5 – Ordre des opérations dans l'algorithme de Doolittle adapté au format de stockage en ligne de ciel.

2. La deuxième modification provient du fait que la matrice stockée est creuse. Un calcul judicieux permet ainsi de ne pas avoir à parcourir l'ensemble des éléments lors du calcul des facteurs. En reprenant les noms des structures du format « skyline » adoptés dans la section précédente, on voit facilement que sur la ligne i, le nombre d'éléments stockés entre le premier élément non nul et la diagonale (non incluse) est donné par :

$$\text{nbi} = \frac{\texttt{diag}(i) - \texttt{diag}(i-1) - 1}{2}, \tag{2.1}$$

et que ces éléments débutent à partir de la colonne $i - \text{nbi}$.

En prenant ces deux remarques en compte, on obtient l'algorithme de Doolittle optimisé pour le format « skyline » . Cette décomposition est faite en notation compacte, *i.e.* les données des facteurs sont écrites dans les mêmes structures de données qui stockent la matrice de départ. Pour la résolution, il suffit d'utiliser les facteurs L et U en effectuant une descente puis une remontée triangulaire :

$$\mathcal{A}\boldsymbol{x} = \boldsymbol{b}(\boldsymbol{p}_F) \Leftrightarrow LU\boldsymbol{x} = \boldsymbol{b}(\boldsymbol{p}_F) \Leftrightarrow \begin{cases} L\boldsymbol{y} = \boldsymbol{b}(\boldsymbol{p}_F)\,; \\ U\boldsymbol{x} = \boldsymbol{y}, \end{cases} \tag{2.2}$$

où $b(\boldsymbol{p}_F) = P_F\boldsymbol{b}$. Encore une fois, l'utilisation de la formule (2.1) permet de réduire considérablement le nombre d'opérations en profitant du format de stockage en ligne

de ciel.

2.2 Méthodes itératives

2.2.1 Algorithmes itératifs classiques

Cette section présente les principaux algorithmes itératifs pour la résolution de systèmes matriciels $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ où $\mathcal{A}$ est une matrice $N \times N$ inversible réelle. Nous discuterons les méthodes suivantes : gradient conjugué (GC), « generalized minimal residual » (GMRES), « biconjugate gradient » (BiCG), « quasi-minimal residual » (QMR), « conjugate gradient squared » (CGS), « stabilized biconjugate gradient » (BiCGSTAB) et « transpose-free quasi-minimal residual » (TFQMR).

Remarque : nous allons parler dans les sections suivantes de préconditionneur. Sans entrer dans les détails, un préconditionneur est une matrice inversible qui permet d'améliorer le conditionnement de la matrice du système linéaire afin de résoudre ce dernier plus rapidement (en moins d'itérations), en particulier lorsque la matrice est mal conditionnée. Si l'on défini deux matrices inversibles d'ordre N, R_1 et R_2, le système linéaire $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ est alors équivalent au système

$$R_1^{-T}\mathcal{A}R_2^{-1}\bar{\boldsymbol{x}} = R_1^{-T}\boldsymbol{b}, \tag{2.3}$$

où $\bar{\boldsymbol{x}} = R_2\boldsymbol{x}$. Nous avons utilisé la notation $R_1^{-T} = (R_1^{-1})^T$. On parle alors de préconditionnement à droite et à gauche. Le but de ces opérations est que les valeurs propres de la nouvelle matrice $R_1^{-T}\mathcal{A}R_2^{-1}$ soient mieux rassemblées que celles de $\mathcal{A}$. Notons qu'il est également possible de n'utiliser qu'un seul préconditionneur G et de résoudre alors le système équivalent

$$G^{-1}\mathcal{A}\boldsymbol{x} = G^{-1}\boldsymbol{b}. \tag{2.4}$$

Le choix des préconditionneurs est généralement dicté par l'application et la structure de la matrice $\mathcal{A}$.

2.2.1.1 Algorithme du gradient conjugué (GC)

Si $\mathcal{A}$ est une matrice symétrique définie positive, le problème $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ revient à résoudre le problème de minimisation : $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ où $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\mathcal{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}$. On construit pour cela, à l'itération k, la solution $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{d}_k$ où $\boldsymbol{d}_k$ est une direction de descente et α_k est un paramètre choisi de sorte que $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$. Ce dernier est défini comme suit :

$$\alpha_k = \arg\min_{\alpha>0} f(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k) = \frac{(\boldsymbol{r}_k, \boldsymbol{r}_k)}{(\mathcal{A}\boldsymbol{d}_k, \boldsymbol{d}_k)}.$$

La méthode du gradient conjugué va construire les vecteurs $\boldsymbol{d}_k$ en s'assurant que

$$(\mathcal{A}\boldsymbol{d}_j, \boldsymbol{d}_i) = 0, \ \forall\, i \neq j. \tag{2.5}$$

De tels vecteurs sont dits conjugués. Cette construction se fait par la récurrence

$$\boldsymbol{d}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k\boldsymbol{d}_k, \tag{2.6}$$

où

$$\boldsymbol{r}_k = \boldsymbol{b} - \mathcal{A}\boldsymbol{x}_k \quad \text{et} \quad \beta_k = \frac{(\mathcal{A}\boldsymbol{d}_k, \boldsymbol{r}_{k+1})}{(\mathcal{A}\boldsymbol{d}_k, \boldsymbol{d}_k)} = \frac{(\boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_k, \boldsymbol{r}_k)}.$$

La récurrence (2.6) assure que la relation (2.5) sera respectée. On prend comme direction initiale $\boldsymbol{d}_0 = \boldsymbol{r}_0 = -\boldsymbol{\nabla} f(\boldsymbol{x}_0)$, ce qui correspond à la direction de la plus forte pente. Notons que les résidus vérifient la récurrence

$$\boldsymbol{r}_{k+1} = \boldsymbol{b} - \mathcal{A}\boldsymbol{x}_{k+1} = \boldsymbol{b} - \mathcal{A}(\boldsymbol{x}_k + \alpha_k\boldsymbol{d}_k) = \boldsymbol{r}_k - \alpha_k\mathcal{A}\boldsymbol{d}_k.$$

Cette série de relations permet ainsi d'obtenir l'algorithme du gradient conjugué. Nous présentons ici la version préconditionnée (cf. algorithme (2.2)). En plus de nécessiter peu d'espace mémoire, on peut montrer que cet algorithme converge en au plus N itérations en arithmétique exacte. En pratique, à cause de l'accumulation d'erreurs numériques, on fixe généralement le nombre d'itérations maximal à $2N$. La limitation principale de l'algorithme est que la matrice $\mathcal{A}$ du système linéaire doit

Algorithme du gradient conjugué préconditionné

Approximation initiale $\boldsymbol{x}_0$, tolérance $\epsilon > 0$, $k = 0$.

Calculer $\boldsymbol{r}_0 = \boldsymbol{b} - \mathcal{A}\boldsymbol{x}_0$,

Résoudre $G\boldsymbol{z}_0 = \boldsymbol{r}_0$,

$\boldsymbol{d}_0 = \boldsymbol{z}_0$.

Tant que $(\boldsymbol{r}_k, \boldsymbol{z}_k) > \epsilon$ faire

$$\alpha_k = \frac{(\boldsymbol{r}_k, \boldsymbol{z}_k)}{(\mathcal{A}\boldsymbol{d}_k, \boldsymbol{d}_k)},$$

$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$,

$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \mathcal{A}\boldsymbol{d}_k$,

Résoudre $G\boldsymbol{z}_{k+1} = \boldsymbol{r}_{k+1}$,

$$\boldsymbol{d}_{k+1} = \boldsymbol{z}_{k+1} + \frac{(\boldsymbol{r}_{k+1}, \boldsymbol{z}_{k+1})}{(\boldsymbol{r}_k, \boldsymbol{z}_k)} \boldsymbol{d}_k,$$

$k \leftarrow k + 1$.

Algorithme 2.2 – Gradient conjugué préconditionné.

être symétrique définie positive.

Afin de présenter les prochaines méthodes itératives, définissons ce qu'est un *sous-espace de Krylov.*

Définition

Le $k^{\text{ème}}$ sous-espace de Krylov $\mathcal{K}_k(\mathcal{A}, \boldsymbol{r})$ associé au vecteur $\boldsymbol{r}$ (non nul) et à la matrice $\mathcal{A}$ est le sous-espace de $\mathbb{R}^N$ de dimension $k < N$ engendré par les vecteurs $\{\boldsymbol{r}, \mathcal{A}\boldsymbol{r}, \ldots, \mathcal{A}^{k-1}\boldsymbol{r}\}$. Lorsqu'il n'y a pas d'ambigüité, on notera $\mathcal{K}_k = \mathcal{K}_k(\mathcal{A}, \boldsymbol{r})$. Notons au passage que $\mathcal{K}_k \subseteq \mathcal{K}_{k+1}$.

Les travaux de Saad (1984) portent sur les méthodes dites de *sous-espaces de Krylov*, en particulier pour des matrices non symétriques. L'idée commune à toutes ces méthodes est de trouver une solution approchée appartenant à l'espace affine $\boldsymbol{x}_0 + \mathcal{K}_k(\mathcal{A}, \boldsymbol{r}_0)$ où le vecteur $\boldsymbol{r}_0$ est le résidu associé à la solution initiale $\boldsymbol{x}_0$: $\boldsymbol{r}_0 = \boldsymbol{b} - \mathcal{A}\boldsymbol{x}_0$. Cette approche se justifie par le résultat suivant (Choquet 1995) :

Théorème

Soit $\mathcal{A}$ une matrice non singulière de taille $N \times N$ et $\boldsymbol{x}^*$ la solution du système $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$. Alors il existe un $k_0 < N$ tel que $\boldsymbol{x}^* \in \boldsymbol{x}_0 + \mathcal{K}_{k_0}(\mathcal{A}, \boldsymbol{r}_0)$.

Remarque : dans les sections qui suivent, nous allons parler à plusieurs reprises de « breakdown » . Ces situations correspondent, en arithmétique exacte, à une division par zéro . En pratique, un « breakdown » se produit lorsque l'on effectue une division par une quantité très petite. Lors de l'implémentation, on essaye de détecter ces situations avant qu'elles ne se produisent. On met alors fin à l'algorithme.

2.2.1.2 Algorithme GMRES

La méthode GMRES (***Generalized Minimal RESidual***) a été proposée par Saad et Schultz (1986). La motivation de ces travaux est qu'il n'y a plus d'interprétation en termes du problème quadratique de la section 2.2.1.1 lorsque la matrice $\mathcal{A}$ n'est pas symétrique. L'algorithme consiste à minimiser, à chaque itération, la norme du résidu associé à la solution $\boldsymbol{x}_0 + \boldsymbol{z}$ où $\boldsymbol{z}$ appartient au sous-espace de Krylov $\mathcal{K}_k(\mathcal{A}, \boldsymbol{r}_0)$. La solution $\boldsymbol{z}_k$ vérifie donc :

$$\boldsymbol{z}_k = \underset{\boldsymbol{z}\in\mathcal{K}_k}{\arg\min} \parallel \boldsymbol{b} - \mathcal{A}(\boldsymbol{x}_0 + \boldsymbol{z}) \parallel_2 . \tag{2.7}$$

On construit alors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ une base orthonormée de l'espace de Krylov en utilisant l'algorithme d'Arnoldi (Saad 1990) qui utilise un processus d'orthogonalisation de type Gram-Schmidt. Définissons la matrice de Hessenberg $\tilde{H}_k$ de taille $(k+1) \times k$:

$$(\tilde{H}_k)_{ij} = \begin{cases} h_{ij} & , \quad 1 \leq j \leq k, \ , \leq i \leq j+1 \ ; \\ 0 & , \quad \text{ailleurs,} \end{cases}$$

où $h_{ij} = (\mathcal{A}\boldsymbol{v}_j, \boldsymbol{v}_i)$. Si V_k est la matrice orthogonale $N \times k$ dont les colonnes sont les vecteurs $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$, alors on peut montrer que (Dutto 1990)

$$\mathcal{A}V_k = V_{k+1}\tilde{H}_k.$$

Si dans l'algorithme d'Arnoldi on choisit $\boldsymbol{v}_1 = \dfrac{\boldsymbol{r}_0}{\parallel \boldsymbol{r}_0 \parallel_2}$, on peut réécrire (2.7) :

$$\begin{aligned}
\parallel \boldsymbol{b} - \mathcal{A}(\boldsymbol{x}_0 + \boldsymbol{z}_k) \parallel_2 &= \min_{\boldsymbol{z}\in\mathcal{K}_k} \parallel \boldsymbol{b} - \mathcal{A}(\boldsymbol{x}_0 + \boldsymbol{z}) \parallel_2 \\
&= \min_{\boldsymbol{z}\in\mathcal{K}_k} \parallel \boldsymbol{r}_0 - \mathcal{A}\boldsymbol{z} \parallel_2 \\
&= \min_{\boldsymbol{y}\in\mathbb{R}^k} \parallel \frac{\boldsymbol{r}_0}{\parallel \boldsymbol{r}_0 \parallel_2} - \mathcal{A}V_k\boldsymbol{y} \parallel_2 \\
&= \min_{\boldsymbol{y}\in\mathbb{R}^k} \parallel \beta\boldsymbol{v}_1 - \mathcal{A}V_k\boldsymbol{y} \parallel_2, \quad \beta = \parallel \boldsymbol{r}_0 \parallel_2 \\
&= \min_{\boldsymbol{y}\in\mathbb{R}^k} \parallel \beta V_{k+1}\boldsymbol{e}_1 - V_{k+1}\tilde{H}_k\boldsymbol{y} \parallel_2, \quad \boldsymbol{e}_1 = (1, 0, \ldots, 0)^T \\
&= \min_{\boldsymbol{y}\in\mathbb{R}^k} \parallel V_{k+1}(\beta\boldsymbol{e}_1 - \tilde{H}_k\boldsymbol{y}) \parallel_2 \\
&= \min_{\boldsymbol{y}\in\mathbb{R}_k} \parallel \beta\boldsymbol{e}_1 - \tilde{H}_k\boldsymbol{y} \parallel_2,
\end{aligned} \tag{2.8}$$

Algorithme GMRES

Approximation initiale $\boldsymbol{x}_0$,

Calculer $\boldsymbol{r}_0 = \boldsymbol{b} - \mathcal{A}\boldsymbol{x}_0$,

$\boldsymbol{v}_1 = \dfrac{\boldsymbol{r}_0}{\| \boldsymbol{r}_0 \|_2}$, $k = 1$.

Tant que la convergence n'est pas atteinte, faire

$h_{ik} = (\mathcal{A}\boldsymbol{v}_k, \boldsymbol{v}_i)$ pour $i = 1, \ldots, k$,

$\tilde{\boldsymbol{v}}_{k+1} = \mathcal{A}\boldsymbol{v}_k - \sum_{i=1}^{k} h_{ik}\boldsymbol{v}_i$,

$h_{k+1,k} = \| \tilde{\boldsymbol{v}}_{k+1} \|_2$,

$\boldsymbol{v}_{k+1} = \dfrac{\tilde{\boldsymbol{v}}_{k+1}}{h_{k+1,k}}$,

Calculer la solution approchée $\boldsymbol{x}_k = \boldsymbol{x}_0 + V_k\boldsymbol{y}_k$ où $\boldsymbol{y}_k$ minimise (2.8),

$k \leftarrow k + 1$.

Algorithme 2.3 – GMRES.

car V_{k+1} est orthogonale.

La minimisation (2.8) peut être effectuée par une méthode de moindres carrés. Cette réécriture permet d'obtenir l'algorithme GMRES 2.3. En arithmétique exacte, la convergence est assurée en au plus N itérations (Saad and Schultz 1984). Cependant, le stockage de V_{k+1} et de $\tilde{H}_k$ peuvent être coûteux lorsque k est grand et on lui préfère alors une autre version moins gourmande en mémoire : l'algorithme GMRES(k) (Saad 1990). Ce dernier consiste à fixer un entier k_0 et à réinitialiser l'algorithme avec la solution approchée $\boldsymbol{x}_{k_0}$. Cette version peut cependant stagner lorsque les redémarrages n'améliorent pas la solution. Il existe également une troisième version de la méthode GMRES qui se prête mieux au préconditionnement. Cette version, nommée DQGMRES, est basée sur un procédé d'orthogonalisation incomplète (Saad and Wu 1995).

2.2.1.3 Algorithme de Lanczos

Cet algorithme (Lanczos 1952) est à la base des algorithmes BiCG, CGS, BiCG-STAB, QMR et TFQMR présentés dans les prochaines sections. On considère cette fois-ci deux espaces de Krylov, associés respectivement à $\mathcal{A}$ et $\mathcal{A}^T$:

$$\begin{cases} \mathcal{K}_k^v &= \mathcal{K}_k(\mathcal{A}, \boldsymbol{r}_0)\,; \\ \mathcal{K}_k^w &= \mathcal{K}_k(\mathcal{A}^T, \boldsymbol{s}_0). \end{cases}$$

On construit alors $\{\boldsymbol{v}_i\}$ et $\{\boldsymbol{w}_i\}$, respectivement des bases de $\mathcal{K}_k^v$ et $\mathcal{K}_k^w$ de sorte que les deux familles soient biorthogonales, *i.e.* :

$$\begin{cases} (\boldsymbol{v}_i, \boldsymbol{w}_j) &= \delta_{ij} \quad , \; 1 \leq i, j \leq k\,; \\ \| \boldsymbol{v}_i \|_2 &= 1 \quad , \; 1 \leq i \leq k\,; \\ \| \boldsymbol{w}_i \|_2 &= 1 \quad , \; 1 \leq i \leq k. \end{cases}$$

On utilise pour cela l'algorithme de biorthogonalisation de Lanczos (Saad 1990).

Remarque : la biorthogonalisation de Lanczos a l'avantage, comparé à l'algorithme d'Arnoldi, de nécessiter un stockage moindre (seulement six vecteurs de taille N, plus le stockage de la matrice T_k qui est définie ci-dessous).

Notons V_k et W_k les matrices dont les colonnes sont les vecteurs de base trouvés. De plus, introduisons la matrice tridiagonale T_k :

$$\begin{pmatrix} \alpha_1 & \beta_2 & 0 & & \dots & 0 \\ \delta_2 & \alpha_2 & \beta_3 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & & \\ 0 & \dots & 0 & \delta_{k-1} & \alpha_{k-1} & \beta_k \\ 0 & & \dots & 0 & \delta_k & \alpha_k \end{pmatrix},$$

où

$$\begin{cases} \alpha_j &= (\mathcal{A}\boldsymbol{v}_j, \boldsymbol{v}_j)\ ; \\ \delta_{j+1} &= |(\mathcal{A}\boldsymbol{v}_j - \alpha_j\boldsymbol{v}_j - \beta_j\boldsymbol{v}_{j-1} \quad , \quad \mathcal{A}^T\boldsymbol{w}_j - \alpha_j\boldsymbol{w}_j - \delta_j\boldsymbol{w}_{j-1})|^{\frac{1}{2}}\ ; \\ \beta_{j+1} &= \dfrac{(\mathcal{A}\boldsymbol{v}_j - \alpha_j\boldsymbol{v}_j - \beta_j\boldsymbol{v}_{j-1} \quad , \quad \mathcal{A}^T\boldsymbol{w}_j - \alpha_j\boldsymbol{w}_j - \delta_j\boldsymbol{w}_{j-1})}{\delta_{j+1}}, \end{cases}$$

on été générés durant l'algorithme de biorthogonalisation de Lanczos. On peut alors montrer les relations suivantes (Saad 2000) :

$$\begin{cases} \mathcal{A}V_k &= V_kT_k + \delta_{k+1}\boldsymbol{v}_{k+1}\boldsymbol{e}_k^T\ ; \\ \mathcal{A}^TV_k &= W_kT_k^T + \beta_{k+1}\boldsymbol{w}_{k+1}\boldsymbol{e}_k^T\ ; \\ W_k^T\mathcal{A}V_k &= T_k. \end{cases} \tag{2.9}$$

où $\boldsymbol{e}_k = (0, \ldots, 0, 1)$. De plus, les $\boldsymbol{w}_i$ générés par l'algorithme de biorthogonalisation de Lanczos vérifient $(\boldsymbol{w}_i, \boldsymbol{r}_k) = 0$, pour $i = 1, 2, \ldots, k$. Ainsi :

$$\begin{aligned} & (\boldsymbol{w}_i, \boldsymbol{r}_k) = 0, \ \ i = 1, \ldots, k \\ \Leftrightarrow\ & W_k^T\boldsymbol{r}_k = 0 \\ \Leftrightarrow\ & W_k^T(\boldsymbol{b} - \mathcal{A}\boldsymbol{x}_k) = 0 \\ \Leftrightarrow\ & W_k^T(\boldsymbol{b} - \mathcal{A}(\boldsymbol{x}_0 + \boldsymbol{z}_k)) = 0 \quad (\boldsymbol{z}_k \in \mathcal{K}_k(\mathcal{A}, \boldsymbol{r}_0)) \\ \Leftrightarrow\ & W_k^T(\boldsymbol{b} - \mathcal{A}(\boldsymbol{x}_0 + V_k\boldsymbol{y}_k)) = 0 \quad (\boldsymbol{y}_k \in \mathbb{R}^k) \\ \Leftrightarrow\ & W_k^T(\boldsymbol{b} - \mathcal{A}\boldsymbol{x}_0 - \mathcal{A}V_k\boldsymbol{y}_k) = 0 \\ \Leftrightarrow\ & W_k^T(\boldsymbol{r}_0 - \mathcal{A}V_k\boldsymbol{y}_k) = 0 \\ \Leftrightarrow\ & W_k^T\boldsymbol{r}_0 = W_k^T\mathcal{A}V_k\boldsymbol{y}_k = T_k\boldsymbol{y}_k \\ \Leftrightarrow\ & W_k^T\beta\boldsymbol{v}_1 = T_k\boldsymbol{y}_k \quad (\beta = \| \boldsymbol{r}_0 \|_2,\ \boldsymbol{v}_1 = \frac{\boldsymbol{r}_0}{\| \boldsymbol{r}_0 \|_2}) \\ \Leftrightarrow\ & W_k^TV_k\beta\boldsymbol{e}_1 = I\beta\boldsymbol{e}_1 = T_k\boldsymbol{y}_k \\ \Leftrightarrow\ & \boldsymbol{y}_k = T_k^{-1}(\beta\boldsymbol{e}_1). \end{aligned}$$

On en déduit ainsi l'algorithme de Lanczos 2.4. L'intérêt de cet algorithme est que

Algorithme de Lanczos

Calculer $\boldsymbol{r}_0 = \boldsymbol{b} - \mathcal{A}\boldsymbol{x}_0$ et $\beta = \| \boldsymbol{r}_0 \|_2$,

Générer les vecteurs $(\boldsymbol{v}_i)$ et $(\boldsymbol{w}_i)$ par la procédure de biorthogonalisation de Lanczos, en initialisant $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$ et en choisissant $\boldsymbol{w}_1$ de façon à avoir $(\boldsymbol{v}_1, \boldsymbol{w}_1) = 1$,

Former la matrice T_k,

Calculer $\boldsymbol{y}_k = T_k^{-1}(\beta \boldsymbol{e}_1)$,

Calculer $\boldsymbol{x}_k = \boldsymbol{x}_0 + V_k \boldsymbol{y}_k$.

Algorithme 2.4 – Lanczos.

l'on résout un système linéaire général en résolvant plusieurs systèmes tridiagonaux plus simples.

Remarque : on peut utiliser pour le calcul de $\boldsymbol{y}_k$ un algorithme adapté aux matrices tridiagonales comme l'algorithme de Thomas (Trefethen and Bau 1997).

2.2.1.4 Algorithme BiCG

La méthode BiCG (***BiConjugate Gradient***) a été proposée par Fletcher (1976). Elle permet de résoudre des systèmes linéaires $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ mettant en jeu des matrices non symétriques. L'algorithme s'inspire de la méthode de Lanczos décrite précédemment. Si on effectue la décomposition LU de T_k,

$$T_k = L_k U_k,$$

et qu'on définit

$$\begin{cases} P_k &= V_k U_k^{-1}\,; \\ Q_k &= W_k L_k^{-T}, \end{cases}$$

on a alors

$$Q_k^T \mathcal{A} P_k = L_k^{-1} W_k^T \mathcal{A} V_k U_k^{-1} = L_k^{-1} T_k U_k^{-1} = I.$$

En se basant sur l'algorithme de Lanczos, et en prenant en compte cette dernière information, on peut en déduire l'algorithme BiCG (Saad 2000). Cet algorithme converge théoriquement en N itérations (Fletcher 1976). Cependant, la norme du résidu peut être irrégulière (Freund et al. 1992) et l'algorithme est sujet à des « breakdown » . L'autre inconvénient majeur est que l'algorithme nécessite des produits matrice-vecteur avec $\mathcal{A}$ et sa transposée $\mathcal{A}^T$.

Remarque : lorsque la matrice $\mathcal{A}$ est symétrique définie positive, l'algorithme BiCG recquiert deux fois plus d'opérations que l'algorithme du gradient conjugué mais lui est théoriquement équivalent (Barrett et al. 1994).

2.2.1.5 Algorithme QMR

La méthode QMR (***Q**uasi-**M**inimal **R**esidual*) (Freund and Nachtigal 1991) est basée sur la méthode de Lanczos décrite précédemment. Alors que BiCG défini des approximations selon une condition de biorthogonalisation, QMR les définit selon une condition de quasi-minimisation.
Reprenons la première équation de (2.9) :

$$\begin{aligned} \mathcal{A} V_k &= V_k T_k + \delta_{k+1} \boldsymbol{v}_{k+1} \boldsymbol{e}_k^T \\ &= V_{k+1} \bar{T}_k, \end{aligned}$$

où

$$\bar{T}_k = \begin{pmatrix} T_k \\ \delta_{k+1} \boldsymbol{e}_k^T \end{pmatrix}.$$

La norme du résidu s'écrit alors :

$$\begin{aligned}
\| \boldsymbol{b} - \mathcal{A}\boldsymbol{x} \|_2 &= \| \boldsymbol{b} - \mathcal{A}(\boldsymbol{x}_0 + V_k\boldsymbol{y}) \|_2 \quad (\boldsymbol{y} \in \mathbb{R}^k) \\
&= \| \boldsymbol{b} - \mathcal{A}\boldsymbol{x}_0 - \mathcal{A}V_k\boldsymbol{y} \|_2 \\
&= \| \boldsymbol{r}_0 - \mathcal{A}V_k\boldsymbol{y} \|_2 \\
&= \| \beta\boldsymbol{v}_1 - V_{k+1}\bar{T}_k\boldsymbol{y} \|_2 \quad (\beta = \| \boldsymbol{r}_0 \|_2,\ \boldsymbol{v}_1 = \frac{\boldsymbol{r}_0}{\beta}) \\
&= \| \beta V_{k+1}\boldsymbol{e}_1 - V_{k+1}\bar{T}_k\boldsymbol{y} \|_2 \\
&= \| V_{k+1}(\beta\boldsymbol{e}_1 - \bar{T}_k\boldsymbol{y}) \|_2 .
\end{aligned}$$

Dans le cas de la méthode GMRES, l'expression pouvait se simplifier car les colonnes de V_{k+1} étaient orthonormales. Ici on va minimiser, à l'aide d'une méthode de moindres carrés, une approximation du résidu :

$$\begin{aligned}
\| \boldsymbol{r}_k \|_2 &= \| V_{k+1}(\beta\boldsymbol{e}_1 - \bar{T}_k\boldsymbol{y}_k) \|_2 \\
&\leq \| V_{k+1} \|_2 \underbrace{\| \beta\boldsymbol{e}_1 - \bar{T}_k\boldsymbol{y}_k \|_2}_{\text{quasi-résidu}}.
\end{aligned}$$

À chaque itération, on aura donc :

$$\begin{cases}
\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \| \beta\boldsymbol{e}_1 - \bar{T}_k\boldsymbol{y}_k \|_2 ; \\
\boldsymbol{x}_k = \boldsymbol{x}_0 + V_k\boldsymbol{y}_k.
\end{cases}$$

On obtient ainsi l'algorithme QMR (Freund and Nachtigal 1994).

2.2.1.6 Algorithme CGS

La méthode CGS (***Conjugate Gradient Squared***) a été proposée par Sonneveld (1989) pour pallier l'un des défauts de la méthode BiCG, en l'occurence l'utilisation de la matrice transposée $\mathcal{A}^T$. Alors que dans l'algorithme BiCG, le résidu $\boldsymbol{r}_j$ et la direction $\boldsymbol{d}_j$ s'écrivent :

$$\boldsymbol{r}_j = \phi_j(\mathcal{A})\boldsymbol{r}_0 \quad \text{et} \quad \boldsymbol{d}_j = \pi_j(\mathcal{A})\boldsymbol{d}_0,$$

où $\phi_j(\mathcal{A})$ et $\pi_j(\mathcal{A})$ sont des polynômes de degré j (avec $\phi_j(0) = 1$), l'algorithme CGS réécrit ces deux vecteurs comme suit :

$$\boldsymbol{r}_j = \phi_j^2(\mathcal{A})\boldsymbol{r}_0 \quad \text{et} \quad \boldsymbol{d}_j = \pi_j^2(\mathcal{A})\boldsymbol{d}_0, \tag{2.10}$$

Cette réécriture permet de s'acquitter de l'utilisation de la matrice $\mathcal{A}^T$ (Saad 1990). En définissant le vecteur auxiliaire $\boldsymbol{q}_j = \phi_{j+1}(\mathcal{A})\pi_j(\mathcal{A})\boldsymbol{r}_0$, on peut montrer que les vecteurs $\boldsymbol{r}_j$, $\boldsymbol{q}_j$ et $\boldsymbol{d}_j$ sont définis par les récurrences suivantes (Saad 2000) :

$$\begin{cases} \boldsymbol{r}_{j+1} &= \boldsymbol{r}_j - \alpha_j\mathcal{A}(2\boldsymbol{r}_j + 2\beta_{j-1}\boldsymbol{q}_{j-1} - \alpha_j\mathcal{A}\boldsymbol{d}_j)\,; \\ \boldsymbol{q}_j &= \boldsymbol{r}_j + \beta_{j-1}\boldsymbol{q}_{j-1} - \alpha_j\mathcal{A}\boldsymbol{d}_j\,; \\ \boldsymbol{d}_{j+1} &= \boldsymbol{r}_{j+1} + 2\beta_j\boldsymbol{q}_j + \beta_j^2\boldsymbol{d}_j. \end{cases}$$

On obtient ainsi l'algorithme CGS (Saad 1990). Cet algorithme converge généralement deux fois plus vite que l'algorithme BiCG et ne nécessite pas de produits matrice-vecteur avec $\mathcal{A}^T$. Cependant, la convergence est irrégulière (Van Der Vorst 1992) et les erreurs d'arrondis et les « overflow » sont plus fréquents car les polynômes ϕ_j et π_j sont mis au carré (cf. équation (2.10)). De plus, l'algorithme est sujet à des « breakdown » .

2.2.1.7 Algorithme BiCGSTAB

La méthode BiCGSTAB (***BiC****onjugate* ***G****radient* ***Stab****ilized*) a été mise au point par Van der Vorst (1992) et possède l'avantage d'avoir une convergence plus régulière que CGS. L'idée consiste à réécrire le résidu $\boldsymbol{r}_j$ et la direction $\boldsymbol{d}_j$:

$$\begin{cases} \boldsymbol{r}_j &= \phi_j(\mathcal{A})\psi_j(\mathcal{A})\boldsymbol{r}_0\,; \\ \boldsymbol{d}_j &= \psi_j(\mathcal{A})\pi_j(\mathcal{A})\boldsymbol{r}_0, \end{cases}$$

où ϕ_j et π_j sont les polynômes associés à la méthode BiCG et ψ_j est un polynôme vérifiant la récurrence

$$\psi_{j+1}(t) = (1 - \omega_j t)\psi_j(t).$$

Le paramètre ω_j est choisi tel que

$$\omega_j = \arg\min \| (I - \omega_j \mathcal{A})\psi_j(\mathcal{A}\phi_{j+1}(\mathcal{A})) \|_2 .$$

En définissant $\boldsymbol{s}_j = \boldsymbol{r}_j - \alpha_j \mathcal{A}\boldsymbol{d}_j$, on peut montrer que (Saad 2000) :

$$\begin{cases} \boldsymbol{r}_{j+1} &= \boldsymbol{r}_j - \alpha_j \mathcal{A}\boldsymbol{d}_j - \omega_j \mathcal{A}\boldsymbol{s}_j \ ; \\ \boldsymbol{x}_{j+1} &= \boldsymbol{x}_j + \alpha_j \boldsymbol{d}_j + \omega_j \boldsymbol{s}_j . \end{cases}$$

On obtient ainsi l'algorithme 2.5. Nous présentons ici la version préconditionnée ($G = R_1^T R_2$ est un préconditionneur, cf. relation (2.3)).
La convergence de cet algorithme est plus régulière que celle des méthodes CGS, QMR et TFQMR, et ne nécessite pas de produits matrice-vecteur avec la matrice transposée $\mathcal{A}^T$. Cependant, l'algorithme perd en précision lors de la mise-à-jour du résidu. Mentionnons le fait que l'échec de CGS entraîne celui de BiCGSTAB.

Remarque : En plus de la matrice G, l'algorithme 2.5 se sert également de la matrice R_1 (requise pour le calcul de ω_k). Pour pallier cet inconvénient, Van der Vorst (1992) propose une alternative en remplaçant ce calcul par

$$\omega_k = \frac{(\mathcal{A}\boldsymbol{s}_k^*, \boldsymbol{s}_k)}{\| \mathcal{A}\boldsymbol{s}_k^* \|_2},$$

qui a l'avantage de ne plus se servir de la matrice R_1.

2.2.1.8 Algorithme TFQMR

La méthode TFQMR (***Transpose-Free Quasi-Minimal Residual***), proposée par Freund (1993), dérive de la méthode CGS (et non de la méthode QMR comme le nom aurait pu le faire penser). Elle se base sur le fait que la solution $\boldsymbol{x}_j$ peut être calculée en deux étapes :

$$\begin{cases} \boldsymbol{x}_{j+\frac{1}{2}} &= \boldsymbol{x}_j + \alpha_j \boldsymbol{u}_j \ ; \\ \boldsymbol{x}_{j+1} &= \boldsymbol{x}_{j+\frac{1}{2}} + \alpha_j \boldsymbol{q}_j . \end{cases}$$

Algorithme BiCGSTAB préconditionné

Approximation initiale $\boldsymbol{x}_0$,

Calculer $\boldsymbol{r}_0 = \boldsymbol{b} - \mathcal{A}\boldsymbol{x}_0$,

Choisir $\boldsymbol{r}_0^*$ tel que $\rho_0 \equiv (\boldsymbol{r}_0, \boldsymbol{r}_0^*) \neq 0$,

$\boldsymbol{d}_0 = \boldsymbol{r}_0, \qquad k = 0.$

Tant que la convergence n'est pas atteinte, faire

Résoudre $G\boldsymbol{d}_k^* = \boldsymbol{d}_k$,

$\alpha_k = \dfrac{\rho_k}{(\mathcal{A}\boldsymbol{d}_k^*, \boldsymbol{r}_0^*)}$,

$\boldsymbol{s}_k = \boldsymbol{r}_k \mathcal{A}\boldsymbol{d}_k^*$,

Résoudre $G\boldsymbol{s}_k^* = \boldsymbol{s}_k$,

$\omega_k = \dfrac{(R_1^{-T}\mathcal{A}\boldsymbol{s}_k^*, R_1^{-T}\boldsymbol{s}_k)}{\| R_1^{-T}\mathcal{A}\boldsymbol{s}_k^* \|_2}$,

$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k^* + \omega_k \boldsymbol{s}_k^*$,

$\boldsymbol{r}_{k+1} = \boldsymbol{s}_k + \omega_k \mathcal{A}\boldsymbol{s}_k^*$,

$\rho_{k+1} = (\boldsymbol{r}_{k+1}, \boldsymbol{r}_0^*)$,

$\beta_k = \dfrac{\rho_{k+1}}{\rho_k} \times \dfrac{\alpha_k}{\omega_k}$,

$\boldsymbol{d}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k(\boldsymbol{d}_k - \omega_k \mathcal{A}\boldsymbol{d}_k^*)$,

$k \leftarrow k + 1$.

Algorithme 2.5 – BiCGSTAB.

Ce « découpage » du calcul implique des changements de notations afin d'éviter des indices multiples de $\frac{1}{2}$. Nous ne nous attarderons pas ici sur les détails des calculs qui aboutissent à l'algorithme. Nous invitons le lecteur à consulter, par exemple, (Saad 2000).

2.2.2 Discussion sur le choix de la méthode itérative

La section précédente nous a permis de passer en revue les principales méthodes itératives de résolution d'un système linéaire. Les méthodes GMRES, GMRES(k), BiCG, QMR et CGS ont été mises de côté pour les raisons suivantes :

- GMRES : demande un stockage mémoire important ;
- GMRES(k) : le moment du redémarrage n'est pas toujours facile à trouver ;
- BiCG : fait appel à la matrice transposée et la convergence du résidu est irrégulière ;
- QMR : fait appel à la matrice transposée ;
- CGS : problèmes d'erreurs d'arrondis et « overflow » .

Les méthodes auxquelles nous nous sommes intéressées dans un premier temps sont par conséquent : GC, BiCGSTAB et TFQMR. Les tableaux 2.1 et 2.2 nous donnent les principales caractéristiques de ces trois méthodes, ainsi que de la méthode GMRES qui est l'un des algorithmes itératifs les plus souvent mentionnés dans la littérature. Le tableau 2.1 nous donne l'espace mémoire nécessaire par itération de chacune des quatres méthodes (*i.e.* le nombre de vecteurs à mémoriser par itération où on ne compte pas la matrice $\mathcal{A}$), tandis que le deuxième tableau nous donne le nombre d'opérations effectuées en termes de produits matrice-vecteur, de produits scalaires et de DAXPY ($\boldsymbol{d} = \alpha\boldsymbol{x} + \boldsymbol{y}$). On constate rapidement que l'algorithme du gradient conjugué est celui qui nécessite le moins d'espace de stockage et le moins d'opérations. Si la matrice du système à résoudre est symétrique et définie positive, ce dernier constitue le meilleur choix. En revanche, pour des matrices non

Tableau 2.1 – Espace mémoire nécessaire à l'itération i pour les principales méthodes itératives.

Méthode	Espace mémoire
GMRES	$(i+5)N$
GC	$6N$
BiCGSTAB	$10N$
TFQMR	$14N$

Tableau 2.2 – Nombre d'opérations à l'itération i pour les principales méthodes itératives.

Méthode	Produits mat./vec.	Produits scalaires	DAXPY
GMRES	1	$i+1$	$i+1$
GC	1	2	3
BiCGSTAB	2	4	6
TFQMR	2	4	10

Tableau 2.3 – Comparatif TFQMR / BiCGSTAB pour $N_A = 30320$.

Algorithme	Produits mat./vec.	Résidu	Stockage	Temps (s)
BiCGSTAB	4909	$0,6298e{-}03$	242560	31,97
TFQMR	90961	$0,8521e{-}01$	333520	712,32

symétriques, la méthode BiCGSTAB nous a semblé être la plus adéquate. En effet, nous avons effectué des tests en résolvant quelques systèmes linéaires avec la matrice A de l'équation (1.43). Ce choix se justifie par le fait que les méthodes projetées nécessitent des produits matrice-vecteur avec A uniquement, qui de plus peut être mal conditionnée pour certains problèmes d'écoulement. Nous avons pour nos tests utilisé les algorithmes de la librairie SPARSKIT (Saad 1990). Cette librairie a été générée à l'aide du compilateur f77 (Absoft - version 9.2) sur un ordinateur Machintosh équipé d'un bi-processeur PowerPC G5 2,5Ghz et doté d'une mémoire vive de 1Go. Les tableaux 2.3 et 2.4 nous montrent un aperçu des résultats obtenus pour des matrices respectivement de taille $N_A = 30320$ et $N_A = 69385$. Précisons que ces matrices sont issues d'un même problème de Poiseuille (cf. section 3.1.2), mais résolu sur des maillages différents. Ces tableaux nous fournissent les informations suivantes : nombre de produits matrice-vecteur, norme du résidu relatif, espace mémoire requis par l'algorithme et temps de résolution (en secondes). Dans l'implémentation des algorithmes de SPARSKIT, l'espace mémoire correspond à la taille d'un tableau de réels qui permet de stocker les vecteurs intermédiaires. Cette taille vaut $8N_A$ pour l'algorithme BiCGSTAB et $11N_A$ pour l'algorithme TFQMR, N_A étant la taille de la matrice A. On constate très clairement que l'algorithme BiCGSTAB nécessite non seulement des besoins moindres en espace mémoire, mais aussi un temps de calcul beaucoup moins important lorsque comparé à la méthode TFQMR.

Tableau 2.4 – Comparatif TFQMR / BiCGSTAB pour $N_A = 69385$.

Algorithme	Produits mat./vec.	Résidu	Stockage	Temps
BiCGSTAB	10149	0,7568e−03	555080	162,52
TFQMR	208155	0,3536e+01	763235	3766,59

Tableau 2.5 – Résultats des algorithmes de Krylov sur le système mixte.

Algorithme	Nombre itér.	Résidu	Mémoire	Temps (s)
BCG	46566	0,1329880e+02	54327	21,95
BiCGSTAB	46567	0,3111281e−04	62088	21,95
TFQMR	46567	0,8147863e−04	85371	23,11
GMRES	46566	0,6374488e+97	132108	28,33
DQGMRES	46566	0,1387398e+95	256177	44,45

2.2.3 Algorithmes projetés

À la section 1.2.4.1, nous avons vu que les discrétisations en temps et en espace des équations pour les écoulements à surfaces libres nous conduisent à des systèmes matriciels de la forme (on considère pour simplifier la présentation les variables primitives $(\vec{U}, \vec{P})$ plutôt que les corrections $(\delta\vec{U}, \delta\vec{P})$) :

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{E} \\ \vec{Q} \end{pmatrix}. \tag{2.11}$$

En appliquant certains algorithmes de Krylov de la librairie SPARSKIT (Saad 1990) sur le système (2.11), nous avons obtenu pour $N = 7761$ le tableau 2.5. Nous nous sommes fixés un nombre d'itérations maximum de $6N = 46566$. Certaines méthodes divergent (GMRES) et d'autres donnent un résidu faible mais avec un nombre d'itérations élevé (BiCGSTAB, TFQMR). Il apparaît ainsi clairement que ces algorithmes ne sont pas adaptés au problème. Plutôt que d'appliquer directement les algorithmes de Krylov sur le système linéaire (Saad 1990), les méthodes projetées

tirent avantage de la structure particulière de la matrice (2.11) en résolvant le système $A\vec{U} = \vec{E}$ dans le noyau de B, puis en effectuant une translation. Deux cas sont alors envisageables :

1. La matrice A est symétrique définie positive. C'est le cas lorsque l'on résout le problème de Stokes avec un seul fluide ;
2. La matrice A est non symétrique. C'est le cas lorsque l'on résout les équations de Navier-Stokes ou lorsque l'on modélise des écoulements mettant en jeu plusieurs fluides.

La version projetée de l'algorithme du gradient conjugué se prête mieux au premier cas. Pour le second, on se tournera plutôt vers l'algorithme BiCGSTAB projeté. Ces deux algorithmes font l'objet des prochaines sections.

2.2.3.1 Algorithme du gradient conjugué projeté préconditionné

L'algorithme PPCG (Gould et al. 1998) (***P**reconditionned **P**rojected **C**onjugate **G**radient*) permet de résoudre des problèmes quadratiques avec contraintes d'égalité de la forme

$$\begin{cases} \min\limits_{\vec{U}} & \frac{1}{2}\vec{U}^T A\vec{U} - \vec{E}^T\vec{U}\ ; \\ \text{s.c.} & B\vec{U} = \vec{Q}, \end{cases} \tag{2.12}$$

où A est une matrice symétrique définie positive de taille $N_A \times N_A$ et B est une matrice de rang maximal de taille $N_B \times N_A$. Comme nous l'avons vu à la section 1.2.5.1, les conditions d'optimalité du problème (2.12) sont :

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{U} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{E} \\ \vec{Q} \end{pmatrix} \Leftrightarrow \begin{cases} A\vec{U} + B^T\vec{P} &= \vec{E}\ ; \\ B\vec{U} &= \vec{Q}, \end{cases} \tag{2.13}$$

où $\vec{P}$ est le vecteur dont les composantes sont les multiplicateurs de Lagrange associés à la contrainte du problème de minimisation (2.13). Ce type de système linéaire apparaît lorsque l'on résout le problème de Stokes avec un seul fluide. Une approche

possible consiste à écrire toute solution $\vec{U}^*$ du problème (2.13) comme :

$$\vec{U}^* = B^T\vec{U}_B^* + Z\vec{U}_Z^*, \tag{2.14}$$

où Z est une matrice de taille $(N_A - N_B) \times N_A$ dont les colonnes forment une base du noyau de B, $\vec{U}_B^* \in \mathbb{R}^{N_B}$ et $\vec{U}_Z^* \in \mathbb{R}^{N_A - N_B}$. La relation (2.13) devient alors :

$$\begin{aligned}
&\begin{cases} A\vec{U} + B^T\vec{P} &= \vec{E}\,; \\ B\vec{U} &= \vec{Q} \end{cases} \\
\Leftrightarrow\ &\begin{cases} A(B^T\vec{U}_B^* + Z\vec{U}_Z^*) + B^T\vec{P} &= \vec{E}\,; \\ B(B^T\vec{U}_B^* + Z\vec{U}_Z^*) &= \vec{Q} \end{cases} \\
\Leftrightarrow\ &\begin{cases} AB^T\vec{U}_B^* + AZ\vec{U}_Z^* + B^T\vec{P} &= \vec{E}\,; \\ BB^T\vec{U}_B^* + BZ\vec{U}_Z^* &= \vec{Q}. \end{cases}
\end{aligned} \tag{2.15}$$

En multipliant la première relation du système (2.15) par Z^T et en prenant en compte le fait que $BZ = 0$ (par définition), on obtient :

$$\begin{aligned}
&\begin{cases} Z^TAB^T\vec{U}_B^* + Z^TAZ\vec{U}_Z^* + Z^TB^T\vec{P} &= Z^T\vec{E}\,; \\ BB^T\vec{U}_B^* + BZ\vec{U}_Z^* &= \vec{Q} \end{cases} \\
\Leftrightarrow\ &\begin{cases} Z^TAZ\vec{U}_Z^* &= -Z^TB^T\vec{P} - Z^TAB^T\vec{U}_B^* + Z^T\vec{E}\,; \\ BB^T\vec{U}_B^* + BZ\vec{U}_Z^* &= \vec{Q} \end{cases} \\
\Leftrightarrow\ &\begin{cases} Z^TAZ\vec{U}_Z^* &= Z^T(\vec{E} - AB^T\vec{U}_B^*)\,; \\ BB^T\vec{U}_B^* &= \vec{Q} \end{cases} \\
\Leftrightarrow\ &\begin{cases} A_{ZZ}\vec{U}_Z^* &= \vec{E}_Z\,; \\ BB^T\vec{U}_B^* &= \vec{Q}, \end{cases}
\end{aligned} \tag{2.16}$$

où $A_{ZZ} = Z^TAZ$ et $\vec{E}_Z = Z^T(\vec{E} - AB^T\vec{U}_B^*)$.

La matrice B étant de rang maximal, et donc BB^t étant inversible, la deuxième relation du système (2.16) nous permet de déduire $\vec{U}_B^*$, de sorte que $\vec{E}_Z$ est connu. Pour le calcul de $\vec{U}_Z^*$, on applique la méthode du gradient conjugué à la première relation du système (2.16). Une version plus pratique s'obtient en incluant la multiplication

par Z et l'addition du terme $B^T\vec{U}_B^*$ (cf. relation (2.14)) à l'intérieur des itérations du gradient conjugué, de sorte que l'on obtient à la sortie de l'algorithme directement la solution $\vec{U}^*$ du problème (2.13). Pour cela, quelques changements de notations sont nécessaires. En définissant

$$P_Z = Z(Z^TZ)^{-1}Z^T,$$

i.e. la matrice de projection orthogonale sur $ker(B)$, on obtient l'algorithme 2.6 non préconditionné (nous reviendrons sur le préconditionnement un peu plus loin dans cette section).

Remarque : l'approximation initiale $\vec{U}_0$ de l'algorithme 2.6 est choisie dans le noyau de B. La solution du système linéaire (2.13) est alors donné par

$$\vec{U} = \vec{U}^* + \vec{U}_B$$

où $\vec{U}^*$ est la solution fournie par l'algorithme et où $\vec{U}_B$ vérifie $B\vec{U}_B = \vec{Q}$. Le vecteur $\vec{U}_B$ est calculée en résolvant le système linéaire :

$$\begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{U}_B \\ \vec{W} \end{pmatrix} = \begin{pmatrix} \vec{0} \\ \vec{Q} \end{pmatrix}, \tag{2.17}$$

où $\vec{W}$ est un vecteur auxiliaire.

Le calcul de la matrice Z est coûteux. Une version plus pratique de l'algorithme consiste donc à se passer de la matrice P_Z en utilisant la forme équivalente :

$$P_B = I - B^T(BB^T)^{-1}B.$$

Algorithme du gradient conjugué projeté - Version 1

Choisir une approximation initiale $\vec{U}_0$ satisfaisant $B\vec{U}_0 = \vec{0}$,

Calculer $\vec{R}_0 = A\vec{U}_0 - \vec{E}$,

$\vec{G}_0 = P_Z\vec{R}_0, \quad \vec{D}_0 = -\vec{G}_0,$

$k = 0.$

Tant que la convergence n'est pas atteinte, faire

$$\alpha_k = \frac{(\vec{R}_k, \vec{G}_k)}{(A\vec{D}_k, \vec{D}_k)},$$

$$\vec{U}_{k+1} = \vec{U}_k + \alpha_k\vec{D}_k,$$

$$\vec{R}^*_k = \vec{R}_k + \alpha_k A\vec{D}_k,$$

$$\vec{G}^*_k = P_Z\vec{R}^*_k,$$

$$\beta_k = \frac{(\vec{R}^*_k, \vec{G}^*_k)}{(\vec{R}_k, \vec{G}_k)},$$

$$\vec{D}_{k+1} = -\vec{G}^*_k + \beta_k\vec{D}_k,$$

$$\vec{G}_{k+1} = \vec{G}^*_k, \quad \vec{R}_{k+1} = \vec{R}^*_k,$$

$k \leftarrow k + 1.$

Algorithme 2.6 – PPCG non préconditionné - version 1.

La matrice P_B n'est pas intéressante en tant que telle car, la matrice BB^T étant dense, le calcul de son inverse est coûteux. En revanche, elle nous permet d'écrire :

$$\begin{aligned} \vec{G}^* &= P_B \vec{R}^* \\ &= (I - B^T(BB^T)^{-1}B)\vec{R}^* \\ &= \vec{R}^* - B^T(BB^T)^{-1}B\vec{R}^* \\ &= \vec{R}^* - B^T\vec{V}^*, \end{aligned} \tag{2.18}$$

où $\vec{V}^*$ vérifie

$$(BB^T)\vec{V}^* = B\vec{R}^*.$$

De plus, nous avons

$$\begin{aligned} B\vec{G}^* &= B\vec{R}^* - BB^T(BB^T)^{-1}B\vec{R}^* \\ &= B\vec{R}^* - B\vec{R}^* \\ &= 0. \end{aligned} \tag{2.19}$$

La combinaison des relations (2.18) et (2.19) nous conduit au système matriciel symétrique indéfini

$$\begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{G}^* \\ \vec{V}^* \end{pmatrix} = \begin{pmatrix} \vec{R}^* \\ \vec{0} \end{pmatrix}. \tag{2.20}$$

La résolution du système (2.20) peut conduire à d'importantes erreurs d'arrondis (Gould et al. 1998). On utilise plutôt l'algorithme itératif 2.7. En définissant l'opérateur de projection

$$\vec{G}^* = P_I(\vec{R}) \Leftrightarrow \begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{G}^* \\ \hat{\vec{G}} \end{pmatrix} = \begin{pmatrix} \vec{R} \\ \vec{0} \end{pmatrix}, \tag{2.21}$$

où $\hat{\vec{G}}$ est un vecteur auxiliaire, on obtient l'algorithme 2.8 non préconditionné.

Algorithme de raffinement itératif

Tant que la convergence n'est pas atteinte, faire

Calculer $\rho_g = \vec{R}^* - \vec{G}^* - B^T\vec{V}^*$, $\rho_v = -B\vec{G}^*$,

Résoudre $\begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \Delta\vec{G}^* \\ \Delta\vec{V}^* \end{pmatrix} = \begin{pmatrix} \rho_g \\ \rho_v \end{pmatrix}$,

Mettre à jour : $\vec{G}^* = \vec{G} + \Delta\vec{G}^*$, $\vec{V}^* = \vec{V}^* + \Delta\vec{V}^*$.

Algorithme 2.7 – Raffinement itératif.

Remarque 1 : l'algorithme 2.8 prend fin lorsque

$$\sqrt{(\vec{R}_k, \vec{G}_k)} < tol,$$

où *tol* est une tolérance fixée par l'utilisateur. Nous reviendrons sur le choix de la tolérance lors de la présentation des résultats numériques, au chapitre 3.

Remarque 2 : l'algorithme 2.8 permet de trouver la solution $\vec{U}$ du système linéaire (2.13). Pour obtenir le vecteur $\vec{P}$, il suffit de résoudre :

$$\begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{\tilde{W}} \\ \vec{P} \end{pmatrix} = \begin{pmatrix} \vec{E} - A\vec{U} \\ \vec{0} \end{pmatrix}, \tag{2.22}$$

où $\vec{\tilde{W}}$ est un vecteur auxiliaire. En effet,

$$\begin{aligned} A\vec{U} + B^T\vec{P} = \vec{E} &\Leftrightarrow BA\vec{U} + BB^T\vec{P} = B\vec{E} \\ &\Leftrightarrow \begin{cases} B\vec{\tilde{W}} + BA\vec{U} + BB^T\vec{P} &= B\vec{E}\,; \\ B\vec{\tilde{W}} &= \vec{0} \end{cases} \\ &\Leftrightarrow \begin{cases} B(\vec{\tilde{W}} + B^T\vec{P}) &= B(\vec{E} - A\vec{U})\,; \\ B\vec{\tilde{W}} &= \vec{0} \end{cases} \\ &\Leftrightarrow \begin{cases} \vec{\tilde{W}} + B^T\vec{P} &= \vec{E} - A\vec{U}\,; \\ B\vec{\tilde{W}} &= \vec{0} \end{cases} \end{aligned} \tag{2.23}$$

Algorithme du gradient conjugué projeté - Version 2

Choisir une approximation initiale $\vec{U}_0$ satisfaisant $B\vec{U}_0 = \vec{0}$,

Calculer $\vec{R}_0 = A\vec{U}_0 - \vec{E}$,

$\vec{R}_0 = P_I(\vec{R}_0), \quad \vec{G}_0 = P_I(\vec{R}_0), \quad \vec{D}_0 = -\vec{G}_0,$

$k = 0.$

Tant que la convergence n'est pas atteinte, faire

$$\alpha_k = \frac{(\vec{R}_k, \vec{G}_k)}{(A\vec{D}_k, \vec{D}_k)},$$

$\vec{U}_{k+1} = \vec{U}_k + \alpha_k \vec{D}_k,$

$\vec{R}^*_k = \vec{R}_k + \alpha_k A\vec{D}_k,$

$\vec{G}^*_k = P_I(\vec{R}^*_k),$

Appliquer l'algorithme de raffinement itératif.

$$\beta_k = \frac{(\vec{R}^*_k, \vec{G}^*_k)}{(\vec{R}_k, \vec{G}_k)},$$

$\vec{D}_{k+1} = -\vec{G}^*_k + \beta_k \vec{D}_k,$

$\vec{G}_{k+1} = \vec{G}^*_k,$

$\vec{R}_{k+1} = \vec{R}^*_k,$

$k \leftarrow k + 1.$

Algorithme 2.8 – PPCG non préconditionné - version 2.

car la matrice B est de rang maximal. La résolution du système (2.22) se fait presque « gratuitement » car on a déjà calculé les facteurs de la matrice de projection. La relation (2.23) s'écrit également

$$BB^T\vec{P} = B(\vec{E} - A\vec{U}),$$

qui est la condition du problème aux moindres carrés linéaires

$$\min_{\vec{P}} \frac{1}{2} \parallel B^T\vec{P} - \vec{E} + A\vec{U} \parallel_2^2 .$$

Remarque 3 : la matrice de projection $P = \begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix}$ joue un rôle important car elle intervient dans le calcul de la solution initiale $\vec{U}_0$ (cf. équation (2.17)), dans le calcul de la projection (2.21) et dans le calcul du vecteur $\vec{P}$ (cf. équation (2.22)). En pratique, on ne va former la matrice de projection qu'une seule fois et la factoriser par un algorithme approprié. La matrice étant symétrique indéfinie, on peut utiliser l'algorithme de factorisation de Bunch-Parlett qui permet d'écrire :

$$P = LB_{BP}L^T, \tag{2.24}$$

où L est une matrice triangulaire inférieure et B_{BP} est une matrice diagonale par blocs, ces derniers étant de taille 1×1 et 2×2. Nous utilisons pour cela les méthodes multifrontales implémentées dans *Ma57* (Harwell 2000). D'un point de vue de la mémoire, on ne stocke finalement que la matrice L, ainsi que l'inverse de la matrice B_{BP}.

Remarque 4 : la version préconditionnée de l'algorithme consiste à remplacer l'opérateur de projection P_I par le suivant :

$$\vec{G}^* = P_G(\vec{R}) \Leftrightarrow \begin{pmatrix} G & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{G}^* \\ \vec{\hat{G}} \end{pmatrix} = \begin{pmatrix} \vec{R} \\ \vec{0} \end{pmatrix},$$

où G est un préconditionneur symétrique, ce qui transforme la projection orthogonale en une projection oblique.

2.2.3.2 Algorithme BiCGSTAB projeté préconditionné

Le raisonnement qui conduit à la version projetée de l'algorithme BiCGSTAB (Orban 2008) est similaire à celui de la section précédente (cf. équations (2.16)). On obtient ainsi l'algorithme 2.9. Ce dernier requiert encore l'utilisation de la matrice Z. Cependant, on peut supposer sans perte de généralité que les colonnes de Z forment une base orthonormale du noyau de B, de sorte que $Z^TZ = I$. La matrice ZZ^T est alors l'opérateur de projection orthogonale sur $ker(B)$. Ainsi,

$$\| Z^TA\vec{S}_k^* \|_2^2 = (\vec{S}_k^*)^TA^TZZ^TA\vec{S}_k^* = (A\vec{S}_k^*)^TP_I(A\vec{S}_k^*) = \| P_I(A\vec{S}_k^*) \|_2^2,$$

ce qui implique que :

$$\omega_k = \frac{(A\vec{S}_k^*, \vec{S}_k^*)}{\| P_I(A\vec{S}_k^*) \|_2^2}. \tag{2.25}$$

Remarque 1 : l'algorithme 2.9 prend fin lorsque

$$\sqrt{(\vec{S}_k, \vec{S}_k^*)} < tol \text{ ou } \sqrt{(\vec{R}_k, P_I(\vec{R}_k))} < tol,$$

où *tol* est une tolérance fixée par l'utilisateur. Nous reviendrons sur le choix de la tolérance lors de la présentation des résultats numériques, au chapitre 3.

Remarque 2 : la version préconditionnée de l'algorithme consiste à remplacer l'opérateur de projection P_I par le suivant :

$$\vec{G}^* = P_G(\vec{R}) \Leftrightarrow \begin{pmatrix} G & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{G}^* \\ \vec{\tilde{G}} \end{pmatrix} = \begin{pmatrix} \vec{R} \\ \vec{0} \end{pmatrix},$$

où G est un préconditionneur symétrique. Toutefois, la matrice $\begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix}$ reste requise pour le calcul de ω_k (cf. équation (2.25)).

Remarque 3 : le système (2.13) peut également être résolu par une méthode de

Algorithme PBCGSTAB

Choisir une approximation initiale $\vec{U}_0$ vérifiant $B\vec{U}_0 = \vec{0}$,

Calculer $\vec{R}_0 = A\vec{U}_0 - \vec{E}$,

$\vec{D}_0 = \vec{R}_0$,

Choisir $\vec{R}_0^*$ tel que $(\vec{R}_0, \vec{R}_0^*) \neq 0$,

$k = 0$.

Tant que la convergence n'est pas atteinte, faire

Calculer $\vec{D}_k^* = P_I(\vec{D}_k)$

$$\alpha_k = \frac{(\vec{R}_k, \vec{R}_0^*)}{(A\vec{D}_k^*, \vec{R}_0^*)},$$

$\vec{S}_k = \vec{R}_k - \alpha_k A\vec{D}_k^*$,

Calculer $\vec{S}_k^* = P_I(\vec{S}_k)$,

$\omega_k = (ZZ^T\vec{S}_k)^T A\vec{S}_k^* / \parallel Z^T A\vec{S}_k^* \parallel_2^2$,

$\vec{U}_{k+1} = \vec{U}_k + \alpha_k\vec{D}_k^* + \omega_k\vec{S}_k^*$,

$\vec{R}_{k+1} = \vec{S}_k - \omega_k A\vec{S}_k^*$,

$$\beta_k = \frac{\alpha_k}{\omega_k} \times \frac{(\vec{R}_{k+1}, \vec{R}_0^*)}{(\vec{R}_k, \vec{R}_0^*)},$$

$\vec{D}_{k+1} = \vec{R}_{k+1} + \beta_k(\vec{D}_k - \omega_k A\vec{D}_k^*)$,

$k \leftarrow k + 1$.

Algorithme 2.9 – PBCGSTAB non préconditionné.

projection orthogonale. Par souci de simplicité, considérons $\vec{Q} = \vec{0}$. On peut alors montrer (Bramley 1992) que la solution $\vec{U}$ peut être obtenue en résolvant le système

$$\mathcal{P}A\mathcal{P}\vec{U} = \mathcal{P}\vec{E}, \tag{2.26}$$

où $\mathcal{P} = I - B^T(BB^T)^{-1}B$ est la matrice de projection sur le noyau de B. La solution $\vec{P}$ peut être déduite en calculant

$$\vec{P} = (B^TB)^{-1}B^T(\vec{E} - A\vec{U}).$$

Tout comme les méthodes projetées, la méthode de projection orthogonale travaille dans le noyau de B. La difficulté de cette dernière réside dans le fait que l'on doit calculer des produits matrice-vecteur avec la matrice $\mathcal{P}A\mathcal{P}$ pour résoudre le système (2.26), ce qui demande la factorisation de Cholesky de BB^T qui est une matrice souvent dense.

2.2.3.3 Espace mémoire et nombre d'opérations des méthodes projetées

Le tableau 2.6 présente l'espace mémoire requis par itération et le nombre d'opérations effectuées par itération des deux algorithmes projetés des sections 2.2.3.1 et 2.2.3.2. Nous avons également inclus dans ce tableau, à titre comparatif, les versions non projetées des algorithmes (*i.e.* GC et BiCGSTAB). Pour les algorithmes projetés, il faut rajouter le stockage des facteurs L et B_{BP} issus de la factorisation de Bunch-Parlett (cf. équation (2.24)) de la matrice de projection, ainsi que la résolution des systèmes linéaires 2.21. Nous traiterons de la performance de ces méthodes projetées au chapitre 3.

Tableau 2.6 – Espace mémoire et nombre d'opérations par itération des algorithmes projetés.

Algorithme	Espace mémoire	Prod. mat./vec.	Prod. scalaires	DAXPY ($\boldsymbol{d} = \alpha\boldsymbol{x} + \boldsymbol{y}$)
PPCG	$7N$	1	3	3
CG	$6N$	1	2	3
PBCGSTAB	$12N$	2	4	6
BiCGSTAB	$10N$	2	4	6

CHAPITRE 3

RÉSULTATS NUMÉRIQUES

Nous présentons dans ce chapitre les résultats numériques obtenus à l'aide des deux algorithmes présentés aux sections 2.2.3.1 et 2.2.3.2. Le premier algorithme, le gradient conjugué projeté préconditionné, nous a permis de traiter des écoulements avec un seul fluide, modélisés par le problème de Stokes. Comme nous l'avons déjà mentionné précédemment, la linéarisation des équations mène à des systèmes linéaires symétriques et l'algorithme PPCG se prête donc bien à leur résolution. Le deuxième algorithme, le BiCGSTAB projeté préconditionné, nous a permis de traiter d'autres types d'écoulements qui mènent à des systèmes linéaires non symétriques. C'est notamment le cas lorsque l'on modélise l'écoulement de deux fluides et/ou lorsque l'on utilise les équations de Navier-Stokes.

Remarque : les résultats ont été générés à l'aide d'un ordinateur Macintosh équipé d'un bi-processeur PowerPC G5 2,5Ghz et doté d'une mémoire vive de 1Go. Nous avons utilisé les compilateurs g95 (GNU - version 4.0.3) et f77/f90 (Absoft - version 9.2).

Remarque : Pour la discrétisation éléments-finis de chacun des domaines, nous avons utilisé l'élément de *Taylor-Hood* (cf. figure 3.1) qui comporte 6 degrés de liberté en vitesse et en pseudo-concentration (aux extrémités et au milieu de chaque arête) et 3 degrés de liberté en pression (aux extrémités de chaque arête).

3.1 Problème de Stokes : écoulement d'un fluide

Nous présentons dans cette section les résultats obtenus sur deux problèmes en régime permanent : l'écoulement de Poiseuille (cf. figure 3.2) et le jet immergé à un fluide (cf. figure 3.3).

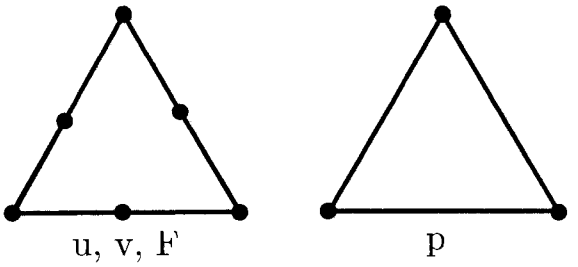


Figure 3.1 – Élément de Taylor-Hood pour la discrétisation éléments-finis.

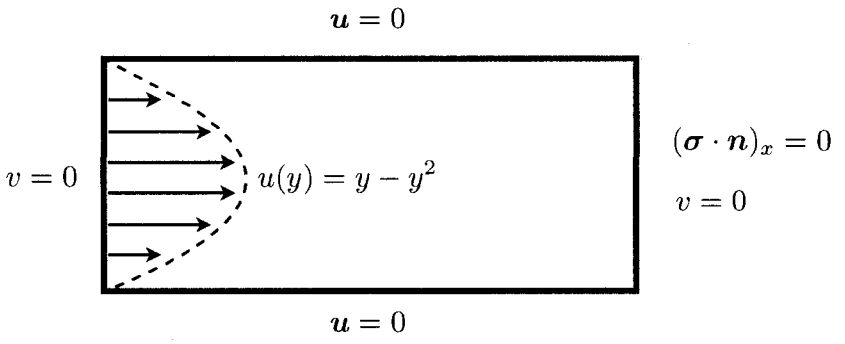


Figure 3.2 – Problème de Poiseuille.

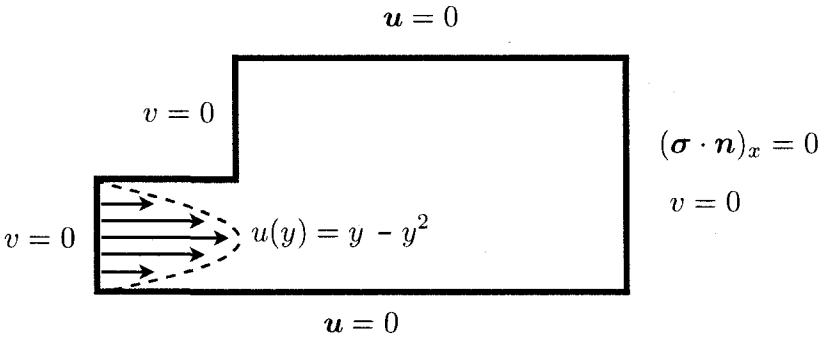


Figure 3.3 – Problème du jet immergé à un fluide.

Les équations de modélisation sont :

$$\begin{cases} -\boldsymbol{\nabla} \cdot (2\mu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})) + \boldsymbol{\nabla} p &= \rho \boldsymbol{f} \ ; \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} &= 0, \end{cases}$$

et leur discrétisation conduit à des systèmes linéaires dans lesquels les matrices sont de la forme

$$\mathcal{A} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \tag{3.1}$$

pour la formulation mixte et de la forme

$$A_r = A + rB^T B \tag{3.2}$$

pour la méthode d'Uzawa. Les matrices A et B sont respectivement de taille $N_A \times N_A$ et $N_B \times N_A$. La matrice $\mathcal{A}$ est de taille $N \times N$ où $N = N_A + N_B$ et la matrice A_r est de taille $N_A \times N_A$. La matrice A étant symétrique définie positive, l'algorithme du gradient conjugué projeté préconditionné (cf. section 2.2.3.1) peut être utilisé pour la formulation mixte. La vitesse est nulle aux bords du domaine de chacune des géométries, sauf en entrée où un profil parabolique en vitesse horizontale $u(y) = y - y^2$ est imposé. Nous comparons, pour chaque problème, les résultats obtenus par :

- la formulation mixte résolue à l'aide de l'algorithme LU ;

- la méthode d'Uzawa résolue à l'aide de l'algorithme LU ;
- la formulation mixte résolue à l'aide de l'algorithme PPCG.

3.1.1 Choix des tolérances

La résolution du problème discret passe par l'utilisation de deux méthodes itératives : l'algorithme de Newton et l'algorithme du gradient conjugué projeté préconditionné. Ces algorithmes prennent fin lorsque certaines quantités (définies respectivement dans les sections 1.2.4 et 2.2.3.1) sont suffisamment petites, *i.e.* inférieures à une tolérance donnée.
Pour la méthode de Newton, nous avons choisi une tolérance fixe égale à 10^{-9}. Pour l'algorithme PPCG, nous avons dans un premier temps essayé une tolérance adaptative :

$$\text{tol} = \| \vec{R} \|_2 * \min(\alpha, \sqrt{\| \vec{R} \|_2}), \tag{3.3}$$

où $\vec{R}$ est le vecteur résidu des équations et $0 < \alpha < 1$. Ainsi, à chaque itération de Newton, la tolérance vaut la norme du résidu des équations multiplié par un petit facteur α lorsque le résidu est grand et par un facteur $\sqrt{\| \vec{R} \|_2}$ lorsque le résidu est petit. Autrement dit, plus on se rapproche de la solution, plus on impose une tolérance serrée. Cette façon de définir la tolérance est classique, mais ne nous a pas fourni de résultats concluants. Nous avons donc opté ici pour une tolérance fixe égale à 10^{-12}.

3.1.2 Problème de Poiseuille

Les tableaux 3.1, 3.2 et 3.3 présentent les résultats numériques obtenus pour le problème de Poiseuille (cf. figure 3.2). Le premier tableau regroupe les résultats provenant de la résolution du problème en formulation mixte par l'algorithme LU (adapté au format de stockage en ligne de ciel, tel qu'expliqué à la section 2.1.2). Ce

tableau nous donne en fonction de N les informations suivantes :

- la densité de la matrice $\mathcal{A}$ (en %) : le rapport entre le nombre d'éléments non nuls de la matrice et le nombre total d'éléments (*i.e.* N^2) : $nnz(\mathcal{A})/N^2$;
- la densité des facteurs (en %) : le rapport entre le nombre d'éléments non nuls des facteurs, *i.e.* celui de L + celui de U, et le nombre total d'éléments de $\mathcal{A}$: $(nnz(L) + nnz(U))/N^2$;
- le temps nécessaire, en secondes, pour calculer la factorisation LU de la matrice $\mathcal{A}$;
- le temps nécessaire, en secondes, pour calculer la remontée/descente triangulaire (cf. équation (2.2)) ;
- le temps total, en secondes, pour résoudre le problème discret (incluant la factorisation, la remontée/descente triangulaire et les autres opérations nécessaires comme l'assemblage du système).

Le second tableau regroupe les résultats provenant de la résolution du problème par la méthode d'Uzawa. Ce tableau nous donne les informations suivantes :

- la densité de la matrice A_r (en %) : le rapport entre le nombre d'éléments non nuls de la matrice et le nombre total d'éléments (*i.e.* N_A^2) : $nnz(A_r)/N_A^2$;
- la densité des facteurs (en %) : le rapport entre le nombre d'éléments non nuls des facteurs, *i.e.* celui de L + celui de U, et le nombre total d'éléments de A_r : $(nnz(L) + nnz(U))/N_A^2$;
- le temps nécessaire, en secondes, pour calculer la factorisation LU de la matrice A_r ;
- le temps nécessaire, en secondes, pour calculer la remontée/descente triangulaire ;
- le temps total, en secondes, pour résoudre le problème discret (incluant la factorisation, la remontée/descente triangulaire et les autres opérations nécessaires comme l'assemblage du système).

Le troisième tableau regroupe les résultats provenant de la résolution des systèmes linéaires en formulation mixte à l'aide de l'algorithme PPCG. On y retrouve les informations suivantes :

- l'ordre de la matrice A (*i.e.* le bloc $(1,1)$ de la matrice $\mathcal{A}$) ;
- le nombre de lignes de la matrice B (*i.e.* le bloc $(2,1)$ de la matrice $\mathcal{A}$) ;
- le nombre d'itérations total effectuées par l'algorithme PPCG ;
- le nombre de produits matrice-vecteur total (rappel : ces produits ne concernent que la matrice A) ;
- la densité de la matrice de projection P (en %) (cf. section 2.2.3.1) ;
- le temps nécessaire, en secondes, pour calculer la factorisation de la matrice de projection P ;
- le temps nécessaire, en secondes, pour calculer la remontée/descente triangulaire ;
- le temps total, en secondes, pour résoudre le problème discret (incluant la factorisation, la remontée/descente triangulaire et les autres opérations nécessaires telle l'assemblage du système).

Nous n'avons pas ici utilisé de préconditionneur, *i.e.* la matrice de projection P est égale à $P = \begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix}$.

Tableau 3.1 – Résultats numériques de la formulation mixte/LU pour le problème de Poiseuille.

Formulation mixte - Algorithme LU					
N	Dens. matrice (%)	Dens. facteurs (%)	Temps fact. (s)	Temps résol. (s)	Temps total (s)
2101	1,27	6,33	0,20	0,00	4,62
8169	0,34	2,74	1,76	0,08	16,17
22592	0,12	1,80	20,12	0,36	67,33
51951	0,06	1,19	125,95	1,50	298,45
80799	0,03	0,95	217.28	2,10	661,32

Tableau 3.2 – Résultats numériques de la formulation Uzawa/LU pour le problème de Poiseuille.

Méthode d'Uzawa - Algorithme LU					
N_A	Dens. matrice (%)	Dens. facteurs (%)	Temps fact. (s)	Temps résol. (s)	Temps total (s)
1829	1,17	5,88	0,09	0,03	4,04
7185	0,31	2,47	0,69	0,03	15,45
19955	0,11	1,68	8,49	0,18	48,21
45993	0,05	1,11	47,16	0,66	142,06
71585	0,03	0,89	111,71	1,26	264,38

Tableau 3.3 – Résultats numériques de la formulation mixte/PPCG pour le problème de Poiseuille.

Formulation mixte - Algorithme PPCG							
N_A	N_B	Itér. PPCG	Prod. mat./vec.	Dens. proj. (%)	Temps fact. (s)	Temps résol. (s)	Temps total (s)
1829	272	207	209	0,44	0,03	0,31	3,53
7185	984	386	388	0,12	0,12	2,78	14,16
19955	2637	658	660	0,04	0,38	13,91	45,30
45993	5958	853	855	0,02	1,04	44,01	117,13
71585	9214	1122	1124	0,01	1,71	90,51	232,05

Pour $N = 80799$, les tableaux 3.4 présentent le détail des itérations de Newton. On y retrouve les informations suivantes :

- la norme euclidienne du vecteur solution ;
- la norme euclidienne du vecteur correction ;
- la norme euclidienne du vecteur résidu des équations ;
- la norme euclidienne du vecteur résidu de l'équation de conservation de la masse $\boldsymbol{\nabla} \cdot \boldsymbol{u}$.

À titre informatif, le tableau 3.5 nous indique, pour chaque taille de matrice N, le nombre d'éléments et le nombre de noeuds du maillage associé.

Tableau 3.4 – Itérations de Newton pour le problème de Poiseuille : $N = 80799$, $N_A = 71585$, $N_B = 214$.

Formulation mixte - Algorithme LU								
Itér. Newton	$\| \vec{U} \|_2$	$\| \delta\vec{U} \|_2$	Résidu équations	Résidu masse	Dens. facteurs (%)	Dens. matrice (%)	Temps fact. (s)	Temps résol. (s)
1	0,0000e+00	0,2249e+03	0,3870e−06	0,2048e−02	0,95	0,03	73,51	0,71
2	0,2249e+03	0,2557e−06	0,2517e−11	0,9218e−07	0,95	0,03	72,12	0,70
3	0,2249e+03	0,1145e−10	0,1232e−13	0,1227e−08	0,95	0,03	71,65	0,70

Méthode d'Uzawa - Algorithme LU								
Itér. Newton	$\| \vec{U} \|_2$	$\| \delta\vec{U} \|_2$	Résidu équations	Résidu masse	Dens. facteurs (%)	Dens. matrice (%)	Temps fact. (s)	Temps résol. (s)
1	0,0000e+00	0,3473e+02	0,2939e−06	0,5257e−09	0,89	0,03	37,22	0,41
2	0,3473e+02	0,6094e−03	0,6995e−08	0,1298e−13	0,89	0,03	37,04	0,43
3	0,3473e+02	0,2476e−03	0,7061e−08	0,2519e−17	0,89	0,03	37,45	0,43

Formulation mixte - Algorithme PPCG									
Itér. Newton	$\| \vec{U} \|_2$	$\| \delta\vec{U} \|_2$	Résidu équations	Résidu masse	Dens. proj. (%)	Nombre itér.	Temps fact. (s)	Temps résol. (s)	Prod. mat./vec.
1	0,0000e+00	0,2249e+03	0,1908e−06	0,9895e−02	0,01	734	0,58	59,37	736
2	0,2249e+03	0,6806e−06	0,1028e−11	0,1228e−07	0,01	381	0,56	30,87	383
3	0,2249e+03	0,9326e−10	0,7276e−13	0,1228e−08	0,01	1	0,57	0,27	3

Tableau 3.5 – Nombre d'éléments et de noeuds des différents maillages utilisés pour le problème de Poiseuille.

N	Nombre d'éléments	Nombre de noeuds
2101	482	1025
8169	1846	3813
22592	5072	10345
51951	11622	23545
80799	18058	36493

3.1.3 Problème du jet immergé à un fluide

Les tableaux 3.6, 3.7 et 3.8 présentent les résultats numériques obtenus pour le problème du jet immergé à un fluide (cf. figure 3.3).

Tableau 3.6 – Résultats numériques de la formulation mixte/LU pour le problème du jet immergé à un fluide.

Formulation mixte - Algorithme LU					
N	Dens. matrice (%)	Dens. facteurs (%)	Temps fact. (s)	Temps résol. (s)	Temps total (s)
4954	0,55	3,18	0,42	0,03	7,80
19347	0,14	1,56	7,80	0,24	44,05
24397	0,11	1,38	12,72	0,32	58,27
39888	0,05	1,09	38,40	0,80	113,83
53759	0,05	0,93	70,52	1,08	179,22
78184	0,04	0,77	141.65	1,32	416,20

Tableau 3.7 – Résultats numériques de la formulation Uzawa/LU pour le problème du jet immergé à un fluide.

Méthode d'Uzawa - Algorithme LU					
N_A	Dens. matrice (%)	Dens. facteurs (%)	Temps fact. (s)	Temps résol. (s)	Temps total (s)
4325	0,55	3,00	0,36	0,04	11,69
17031	0,13	1,48	4,44	0,20	46,59
21513	0,10	1,29	6,96	0,24	60,03
35233	0,06	1,05	23,28	0,52	117,39
47525	0,05	0,89	41,84	0,80	163,24
69185	0,03	0,74	92,96	1,35	268,18

Tableau 3.8 – Résultats numériques de la formulation mixte/PPCG pour le problème du jet immergé à un fluide.

Formulation mixte - Algorithme PPCG							
N_A	N_B	Itér. PPCG	Prod. mat./vec.	Dens. proj. (%)	Temps fact. (s)	Temps résol. (s)	Temps total (s)
4325	629	217	223	0,17	0,06	0,81	8,31
17031	2316	409	415	0,05	0,30	6,98	34,29
21513	2884	502	508	0,04	0,39	10,98	45,64
35233	4655	582	588	0,02	0,69	23,52	81,36
47525	6234	655	671	0,02	0,97	35,43	111,25
69185	8999	839	845	0,01	1,59	69,47	178,51

Pour $N = 78184$, les tableaux 3.9 présentent le détail des itérations de Newton. À titre informatif, le tableau 3.10 nous indique, pour chaque taille de matrice N, le nombre d'éléments et le nombre de noeuds du maillage associé.

Tableau 3.9 – Itérations de Newton pour le problème du jet immergé à un fluide : $N = 78184$, $N_A = 69185$, $N_B = 8999$.

Formulation mixte - Algorithme LU							
Itér. Newton	$\| \vec{U} \|_2$	$\| \delta\vec{U} \|_2$	Résidu équations	Résidu masse	Dens. facteurs (%)	Dens. matrice (%)	Temps fact. (s)	Temps résol. (s)
1	0.0000e+00	0,1531e+03	0,2043e−06	0,2680e+01	0,77	0,03	47,29	0,44
2	0,1531e+03	0,3969e−05	0,4016e−13	5,6161e−08	0,77	0,03	47,12	0,44
3	0,1531e+03	0,2873e−08	0,7309e−14	2,9660e−14	0,77	0,03	47,24	0,44

Méthode d'Uzawa - Algorithme LU								
Itér. Newton	$\| \vec{U} \|_2$	$\| \delta\vec{U} \|_2$	Résidu équations	Résidu masse	Dens. facteurs (%)	Dens. matrice (%)	Temps fact. (s)	Temps résol. (s)
1	0,0000e+00	0,1888e+02	0,1452e−06	0,7213e−08	0,74	0,03	23,01	0,34
2	0,1888e+02	0,1667e−02	0,3697e−08	0,4031e−12	0,74	0,03	23,53	0,34
3	0,1889e+02	0,1894e−06	0,3697e−08	0,3838e−16	0,74	0,03	23,17	0,34
4	0,1889e+02	0,9754e−08	0,3695e−08	0,8623e−17	0,74	0,03	23,25	0,33

Formulation mixte - Algorithme PPCG									
Itér. Newton	$\| \vec{U} \|_2$	$\| \delta\vec{U} \|_2$	Résidu équations	Résidu masse	Dens. proj. (%)	Nombre itér.	Temps fact. (s)	Temps résol. (s)	Prod. mat./vec.
1	0,0000e+00	0,1531e+03	0,9408e−07	0,6929e−01	0,01	593	0,52	49,24	595
2	0,1531e+03	0,2047e−04	0,9500e−12	0,8300e−08	0,01	246	0,53	20,04	248
3	0,1531e+03	0,3247e−09	0,9599e−14	0,7553e−15	0,01	0	0,53	0,19	2

Tableau 3.10 – Nombre d'éléments et de noeuds des différents maillages utilisés pour le problème du jet immergé à un fluide.

N	Nombre d'éléments	Nombre de noeuds
4954	1127	2354
19347	4446	9091
24397	5462	11145
39888	8924	18129
53759	12071	24470
78184	17459	35314

3.1.4 Commentaires à propos des résultats

Les résultats des tableaux 3.4 et 3.9 nous amènent à faire les commentaires suivants :

1. les normes des vecteurs solutions des formulations mixte/LU et mixte/PPCG sont identiques, ce qui nous conforte dans le fait que les solutions calculées sont les mêmes ;
2. la divergence discrète est plus petite pour la formulation Uzawa/LU mais le résidu des équations est plus important ;
3. l'algorithme PPCG effectue très peu d'itérations en comparaison avec sa limite théorique qui est la taille du système linéaire. Par exemple, en se basant sur les données du tableau 3.3, pour $N_A = 71585$, l'algorithme PPCG effectue 1122 itérations plutôt que 80799.

Pour nous assurer de la validité de nos résultats, nous avons calculé, pour chacun des problèmes, la différence entre les solutions obtenues par chacune des méthodes. Nous avons ainsi pu vérifier que nous obtenions à chaque fois une différence presque nulle (de l'ordre de 10^{-14} pour la vitesse et 10^{-12} pour la pression).
Pour la comparaison des algorithmes, nous avons considéré les deux critères usuels que sont le temps d'exécution et l'espace mémoire utilisé. Les méthodes directes de type LU nécessitent le stockage de la matrice entière. Même si des structures de données adaptées, comme celle en ligne de ciel, permettent de réduire les besoins en mémoire lorsque la matrice est creuse, ces derniers restent cependant importants. L'un des avantages de la méthode PPCG, et de la plupart des méthodes itératives en général, est qu'elle ne nécessite pas le stockage de la matrice entière. En effet, l'algorithme nécessite :

- le stockage de B pour former la matrice de projection P. Une fois cette dernière formée, B n'a plus besoin d'être gardée en mémoire ;
- des produits matrice-vecteur avec A. Il n'est pas nécessaire de former la matrice.

Pour le problème de Poiseuille, selon les données fournies par les tableaux 3.1, 3.2 et 3.3, nous avons pour $N = 80799$:

- pour la formulation mixte/LU : densité de la matrice $= 0,95\%$;
- pour la formulation Uzawa/LU : densité de la matrice $= 0,89\%$;
- pour la formulation mixte/PPCG : densité de la matrice de projection $= 0,01\%$.

Pour le problème du jet immergé à un fluide nous avons, suivant les données des tableaux 3.6, 3.7 et 3.8 pour $N = 78184$:

- pour la formulation mixte/LU : densité de la matrice $= 0,77\%$;
- pour la formulation Uzawa/LU : densité de la matrice $= 0,74\%$;
- pour la formulation mixte/PPCG : densité de la matrice de projection $= 0,01\%$.

On constate donc clairement que les besoins en mémoire requis par l'algorithme PPCG sont beaucoup moins importants que ceux requis par l'algorithme LU, que ce soit en formulation mixte ou par la méthode d'Uzawa. Pour la comparaison des temps d'exécution, nous pouvons nous reporter aux figures 3.4 et 3.5 pour constater non seulement que l'algorithme PPCG est plus rapide, mais que cette rapidité s'accentue lorsque la taille du problème augmente. Ces graphes nous montrent également que le temps de l'algorithme PPCG est presque linéaire en N.

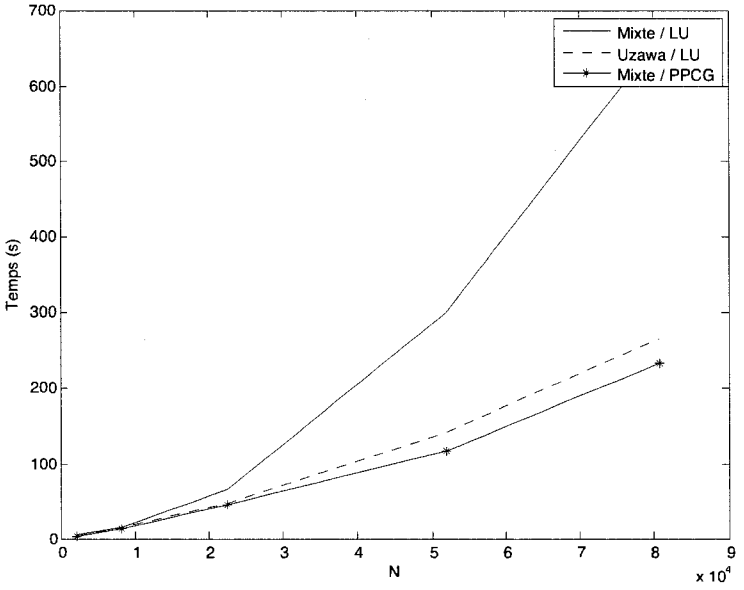


Figure 3.4 – Comparaison des temps de résolution du problème discret pour le problème de Poiseuille.

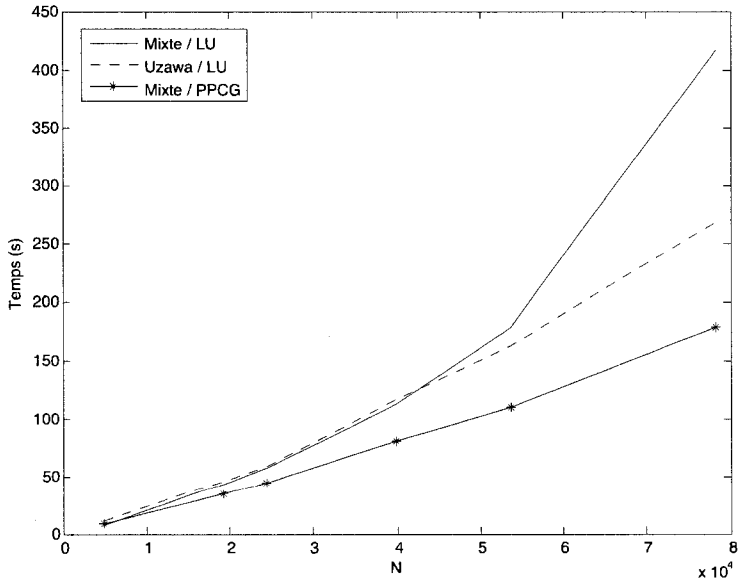
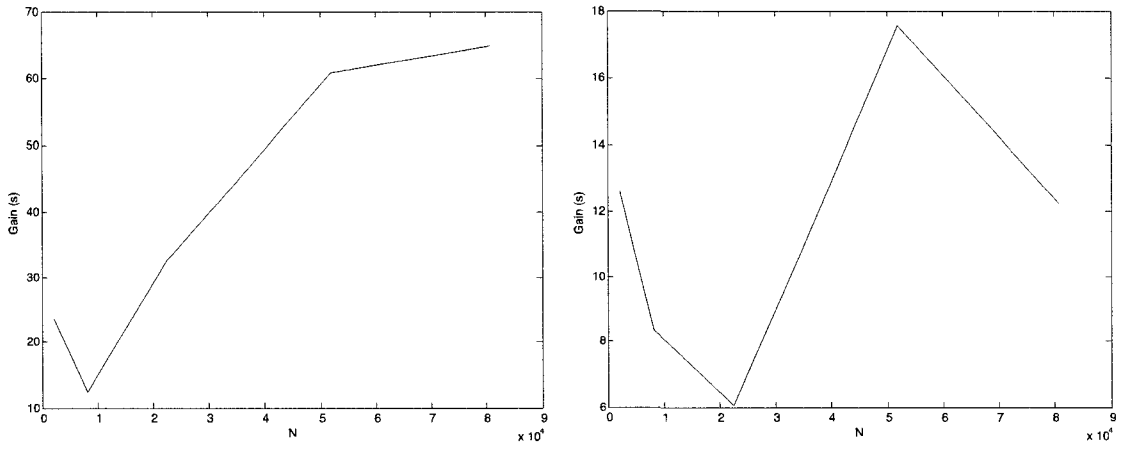


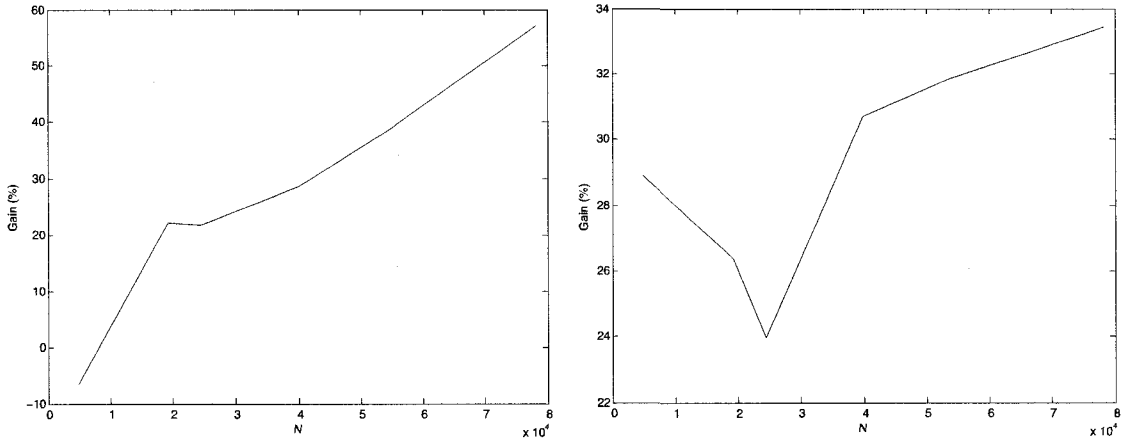
Figure 3.5 – Comparaison des temps de résolution du problème discret pour le problème du jet immergé à un fluide.

Les figures 3.6 et 3.7 montrent l'évolution du gain en temps de la formulation mixte/PPCG par rapport aux autres méthodes en fonction de la taille du problème.



(a) Par rapport à la formulation mixte/LU (b) Par rapport à la formulation Uzawa/LU

Figure 3.6 – Gain en temps de la formulation mixte/PPCG pour le problème de Poiseuille.



(a) Par rapport à la formulation mixte/LU (b) Par rapport à la formulation Uzawa/LU

Figure 3.7 – Gain en temps de la formulation mixte/PPCG pour le problème du jet immergé à un fluide.

3.2 Allée de Von Karman avec surface libre

Nous présentons dans cette section les résultats numériques obtenus pour le problème de l'allée de Von Karman avec une surface libre. Un objet est immergé dans un liquide, près d'une surface libre qui sépare ce liquide d'un gaz. En imposant une vitesse horizontale constante en haut, en bas et à l'entrée du domaine (cf. figure 3.8), on observe autour de l'obstacle la naissance d'une allée dite de Von Karman (cf. figure 3.9).

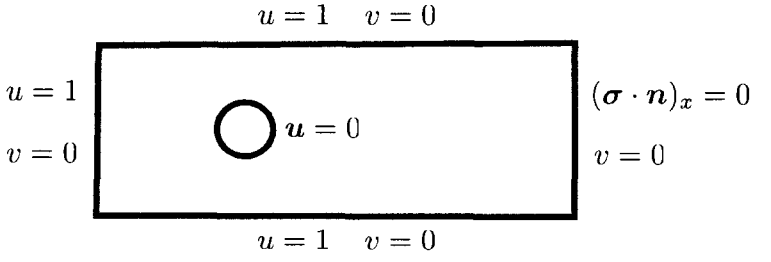


Figure 3.8 – Problème de l'allée de Von Karman.

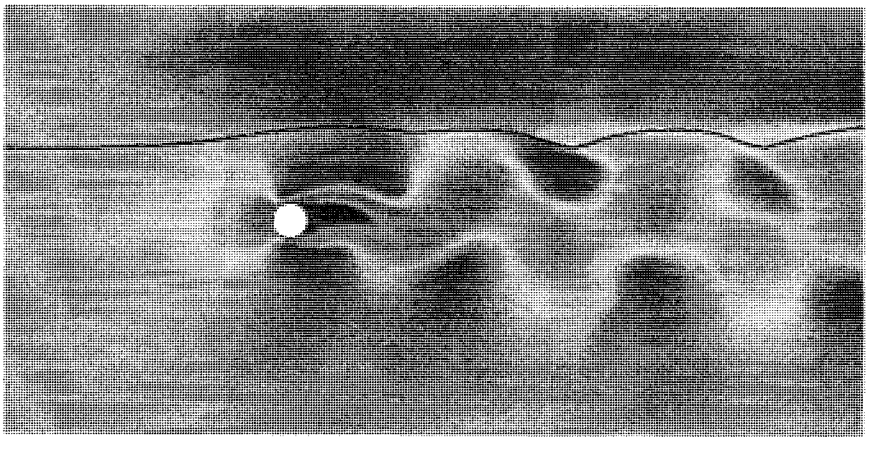


Figure 3.9 – Allée de Von Karman avec une surface libre.

3.2.1 Adimensionnalisation

La dynamique de cet écoulement est caractérisée par un nombre adimensionnel, le *nombre de Reynolds*, qui quantifie l'importance des effets visqueux. Ce nombre, noté Re, apparaît lors de l'adimensionnalisation des équations de modélisation. Reprenons les équations (1.16) obtenues à la section 1.1.1 :

$$\begin{cases} \rho(\dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}) - \boldsymbol{\nabla}\cdot(2\mu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})) + \boldsymbol{\nabla}p &= \rho\boldsymbol{f}\ ; \\ \dfrac{\partial F}{\partial t} + \boldsymbol{u}\cdot\boldsymbol{\nabla}F &= 0\ ; \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} &= 0. \end{cases}$$

Considérons le cas de deux fluides et définissons les quantités de référence du tableau 3.11.

Tableau 3.11 – Paramètres pour l'adimensionnalisation.

L_0	diamètre du canal en sortie
U_0	vitesse moyenne du fluide en entrée
ρ_0	densité du liquide (fluide 2)
μ_0	viscosité du liquide (fluide 2)
p_0	pression de référence
f_0	force de référence

En posant $\tilde{\boldsymbol{u}} = \dfrac{\boldsymbol{u}}{U_0}$, $\tilde{\boldsymbol{x}} = \dfrac{\boldsymbol{x}}{L_0}$, $\tilde{\rho} = \dfrac{\rho}{\rho_0}$ et $\tilde{\mu} = \dfrac{\mu}{\mu_0}$, on obtient pour l'équation de

conservation de la quantité de mouvement :

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho(\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u} - \boldsymbol{\nabla}\cdot(2\mu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})) + \boldsymbol{\nabla}p = \rho\boldsymbol{f}$$

$$\Leftrightarrow$$

$$\tilde{\rho}\rho_0\frac{U_0}{t_0}\frac{\partial \tilde{\boldsymbol{u}}}{\partial \tilde{t}} + \tilde{\rho}\rho_0\frac{U_0^2}{L_0}(\tilde{\boldsymbol{u}}\cdot\tilde{\boldsymbol{\nabla}})\tilde{\boldsymbol{u}} - \frac{1}{L_0}\tilde{\boldsymbol{\nabla}}\cdot(2\tilde{\mu}\mu_0\frac{1}{L_0}\tilde{\dot{\boldsymbol{\gamma}}}(U_0\tilde{\boldsymbol{u}})) + \frac{p_0}{L_0}\tilde{\boldsymbol{\nabla}}\tilde{p} = \tilde{\rho}\rho_0\boldsymbol{f}$$

$$\Leftrightarrow$$

$$\tilde{\rho}\rho_0\frac{U_0}{t_0}\frac{\partial \tilde{\boldsymbol{u}}}{\partial \tilde{t}} + \tilde{\rho}\rho_0\frac{U_0^2}{L_0}(\tilde{\boldsymbol{u}}\cdot\tilde{\boldsymbol{\nabla}})\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{\nabla}}\cdot(2\frac{\tilde{\mu}\mu_0 U_0}{L_0^2}\tilde{\dot{\boldsymbol{\gamma}}}(\tilde{\boldsymbol{u}})) + \frac{p_0}{L_0}\tilde{\boldsymbol{\nabla}}\tilde{p} = \tilde{\rho}\rho_0\boldsymbol{f}$$

$$\Leftrightarrow$$

$$\tilde{\rho}\frac{L_0}{t_0U_0}\frac{\partial \tilde{\boldsymbol{u}}}{\partial \tilde{t}} + \tilde{\rho}(\tilde{\boldsymbol{u}}\cdot\tilde{\boldsymbol{\nabla}})\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{\nabla}}\cdot(2\frac{\tilde{\mu}\mu_0}{L_0U_0\tilde{\rho}\rho_0}\tilde{\dot{\boldsymbol{\gamma}}}(\tilde{\boldsymbol{u}})) + \frac{p_0}{\rho\rho_0U_0^2}\tilde{\boldsymbol{\nabla}}\tilde{p} = \frac{L_0}{U_0^2}\boldsymbol{f}$$

$$\Leftrightarrow$$

$$\tilde{\rho}\frac{L_0}{t_0U_0}\frac{\partial \tilde{\boldsymbol{u}}}{\partial \tilde{t}} + \tilde{\rho}(\tilde{\boldsymbol{u}}\cdot\tilde{\boldsymbol{\nabla}})\tilde{\boldsymbol{u}} - \frac{1}{Re}\tilde{\boldsymbol{\nabla}}\cdot(2\tilde{\mu}\tilde{\rho}\rho_0\tilde{\dot{\boldsymbol{\gamma}}}(\tilde{\boldsymbol{u}})) + \frac{p_0}{\rho_0U_0^2}\tilde{\boldsymbol{\nabla}}\tilde{p} = \tilde{\rho}\frac{L_0}{U_0^2}\boldsymbol{f}$$

où $Re = \dfrac{U_0L_0\rho_0}{\mu_0}$ est le nombre de Reynolds.

Finalement, en prenant $p_0 = \rho_0U_0^2$, $f_0 = \dfrac{U_0^2}{L_0}$ et $t_0 = \dfrac{L_0}{U_0}$, on obtient :

$$\tilde{\rho}\frac{\partial \tilde{\boldsymbol{u}}}{\partial \tilde{t}} + \tilde{\rho}(\tilde{\boldsymbol{u}}\cdot\tilde{\boldsymbol{\nabla}})\tilde{\boldsymbol{u}} - \frac{1}{Re}\tilde{\boldsymbol{\nabla}}\cdot(2\tilde{\mu}\tilde{\dot{\boldsymbol{\gamma}}}(\tilde{\boldsymbol{u}})) + \tilde{\boldsymbol{\nabla}}\tilde{p} = \tilde{\rho}\tilde{\boldsymbol{f}}.$$

Pour alléger les équations, nous enlèverons par la suite la notation « ˜ » .

Remarque : le nombre de Reynolds est sans dimension :

$$Re = \frac{U_0L_0\rho_0}{\mu_0} \equiv \frac{(m.s^{-2})\cdot(m)\cdot(kg.m^{-3})}{(Pa.s)} \equiv \frac{(m.s^{-2})\cdot(m)\cdot(kg.m^{-3})}{kg.m.s^{-2}.m^{-2}.s} \equiv 1.$$

L'adimensionnalisation des équations de conservation de la masse et de transport de la pseudo-concentration s'effectue de façon similaire.

3.2.2 Choix des tolérances

La résolution du problème discret passe par l'utilisation de deux méthodes itératives, l'algorithme de Newton et l'algorithme PBCGSTAB. Ces algorithmes prennent fin lorsque certaines quantités (définies respectivement dans les sections 1.2.4 et 2.2.3.2) sont suffisamment petites, *i.e.* inférieures à une certaine tolérance. Pour la méthode de Newton, nous avons choisi une tolérance fixe égale à 10^{-6}. Pour l'algorithme PBCGSTAB, nous avons choisi une tolérance adaptative telle que définie par l'équation (3.3), la tolérance fixe n'ayant pas fourni de résultats satisfaisants. Nous avons choisi de prendre, après quelques tests, $\alpha = 0,1$. Le tableau 3.12 nous montre l'évolution de la tolérance en fonction du résidu pour le premier pas de temps.

Tableau 3.12 – Évolution de la tolérance en fonction du résidu des équations.

$\| \vec{R} \|_2$	$\min(0,1; \sqrt{\| \vec{R} \|_2})$	tol
0,5504e+00	0,1000e+00	0,5504e−01
0,5778e−01	0,1000e+00	0,5778e−02
0,6986e−02	0,8358e−01	0,5838e−03
0,7050e−03	0,2655e−01	0,1872e−04
0,7306e−04	0,8547e−02	0,6244e−06
0,7545e−05	0,2746e−02	0,2072e−07

3.2.3 Résultats numériques

L'allée de Von Karman n'apparaît que pour une certaine plage de valeurs du nombre de Reynolds. Nous avons choisi ici $Re = 100$. Le problème étant transitoire, on résout le système linéaire (1.43) à chaque pas de temps. La matrice A est cette fois-ci non symétrique. En effet, comparativement aux problèmes précédents, on rajoute :

- le terme de convection $\rho(\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}$;
- l'équation de transport de la pseudo-concentration ;

- les termes transitoires $\rho\dfrac{\partial \boldsymbol{u}}{\partial t}$ et $\dfrac{\partial F}{\partial t}$.

L'algorithme du gradient conjugué projeté préconditionné n'est donc pas approprié pour ce problème. Nous avons donc utilisé l'algorithme PBCGSTAB présenté à la section 2.2.3.2. Pour nos tests nous avons choisi un pas de temps $\Delta t = 0,1$ et nous nous sommes fixés une limite de 800 pas de temps.

Nous avons compilé dans les tableaux 3.13, 3.14 et 3.15 les résultats obtenus pour quelques pas de temps. Le maillage utilisé comporte 18950 éléments et 38144 noeuds. Les matrices assemblées résultantes sont de taille $N = 123215$. Ces tableaux nous donnent les informations suivantes en fonction du pas de temps :

- le nombre d'itérations de Newton ;
- le temps nécessaire, en secondes, pour calculer les différentes factorisations matricielles. Ce temps représente :
  1. le temps de factorisation LU de la matrice $\mathcal{A}$ pour la formulation mixte/LU ;
  2. le temps de factorisation LU de la matrice A_r pour la formulation Uzawa/LU ;
  3. le temps de factorisation de la matrice de projection P pour la formulation mixte/PBCGSTAB.
- le temps nécessaire, en secondes, pour calculer la remontée/descente triangulaire associée à la factorisation du système (1.43) ;
- le temps nécessaire, en secondes, pour résoudre le problème discret à un pas de temps donné ;
- le temps cumulatif, en secondes, nécessaire pour la résolution du problème discret jusqu'à un pas de temps donné.

Les tableaux 3.16 présentent le détail des itérations de Newton pour le dernier pas de temps.

Tableau 3.13 – Temps de calcul de la formulation mixte/LU pour le problème de l'allée de Von Karman avec une surface libre.

Pas de temps	Itér. Newton	Temps fact.(s)	Temps Résol. (s)	Temps itér. Newton (s)	Temps cumulatif (s)
100	5	327,29	2,10	401,61	42236,10
200	5	336,42	2,16	411,84	82395,24
300	5	326,60	2,12	400,94	122623,85
400	5	328,13	2,18	402,83	162742,94
500	5	326,63	2,10	400,95	203035,41
600	5	326,60	2,12	400,97	243233,51
700	5	326,99	2,08	401,41	283502,50
800	5	327,73	2,12	402,30	323832,21

Tableau 3.14 – Temps de calcul de la formulation Uzawa/LU pour le problème de l'allée de Von Karman avec une surface libre.

Pas de temps	Itér. Newton	Temps fact.(s)	Temps Résol. (s)	Temps itér. Newton (s)	Temps cumulatif (s)
100	5	220,33	1,68	304,41	32605,48
200	5	220,30	1,66	304,37	63050,92
300	5	229,56	1,70	315,43	93882,68
400	5	220,39	1,68	304,49	124729,99
500	5	220,82	1,67	304,93	155272,28
600	5	220,31	1,65	304,40	185719,05
700	5	220,25	1,68	304,32	216326,71
800	5	220,27	1,69	304,35	246774,29

Tableau 3.15 – Temps de calcul de la formulation mixte/PBCGSTAB pour le problème de l'allée de Von Karman avec une surface libre.

Pas de temps	Prod. mat./vec.	Itér. Newton	Temps fact.(s)	Temps Résol. (s)	Temps itér. Newton (s)	Temps cumulatif (s)
100	88	5	4,06	8,60	72,90	7814,06
200	150	6	5,10	14,87	93,46	15979,13
300	122	6	4,95	12,86	91,30	25232,75
400	122	6	5,08	12,21	90,77	34435,58
500	134	6	5,13	13,04	91,71	43632,77
600	116	6	4,98	12,64	91,36	52780,54
700	118	6	4,99	12,61	91,32	61898,54
800	126	6	5,10	12,99	91,72	70974,53

3.2.4 Commentaires à propos des résultats

À la vue des résultats des tableaux 3.13, 3.14 et 3.15, il est clair que la formulation mixte/PBCGSTAB nous donne de meilleurs résultats en terme de temps de calcul. Ce constat est confirmé par la figure 3.10 qui représente le temps cumulatif de calcul en fonction du pas de temps pour chacune des trois formulations.

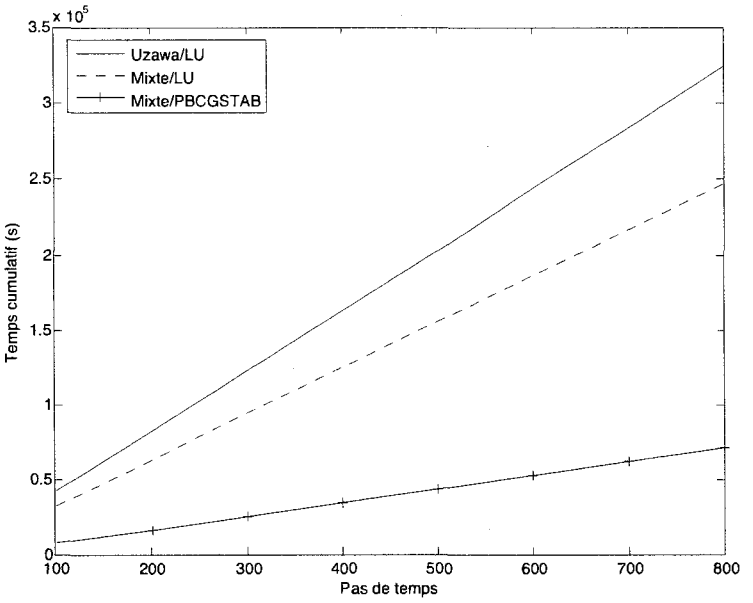
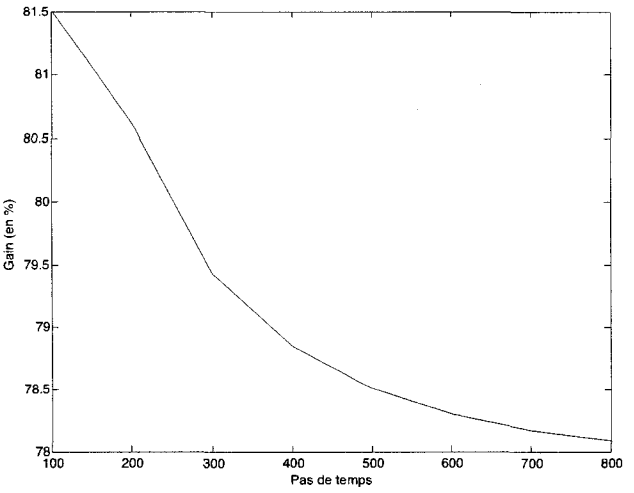


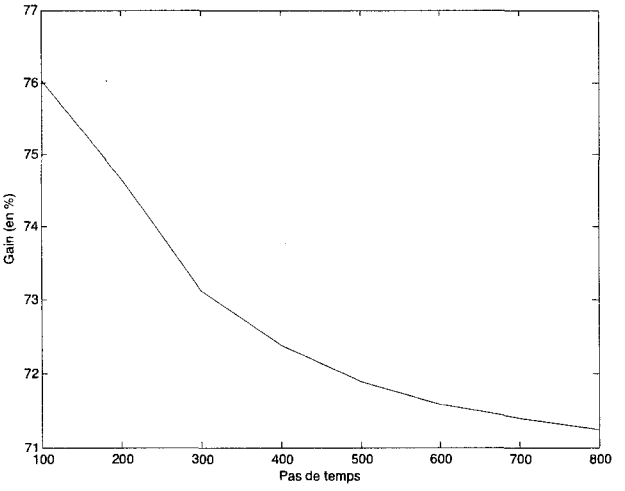
Figure 3.10 – Évolution du temps cumulatif de calcul en fonction du pas de temps pour le problème de l'allée de Von Karman avec une surface libre.

Les figures 3.11(a) et 3.11(b) nous montrent quant à elles le gain en temps (en %) de la formulation mixte/PBCGSTAB par rapport aux deux autres formulations. Ce gain se situe entre 70% et 75% par rapport à la formulation Uzawa/LU et entre 78% et 81.5% par rapport à la formulation mixte/LU.

Remarque : l'utilisation de la tolérance adaptative se traduit dans le troisième tableau du tableau 3.16 par une augmentation du nombre de produits matrice-vecteur effectués par l'algorithme PBCGSTAB au fil des itérations de Newton. Cette tendance se confirme à chaque pas de temps (cf. tableau 3.17).



(a) Par rapport à la formulation mixte/LU



(b) Par rapport à la formulation Uzawa/LU

Figure 3.11 – Gain en temps de calcul de la formulation mixte/PBCGSTAB pour le problème de l'allée de Von Karman avec une surface libre.

Tableau 3.16 – Itérations de Newton pour le problème de l'allée de Von Karman avec une surface libre : $N = 62966$, $N_A = 58019$, $N_B = 4947$.

Formulation mixte - Algorithme LU								
Itér. Newton	$\| \vec{U} \|_2$	$\| \delta\vec{U} \|_2$	Résidu équations	Résidu masse	Dens. facteurs (%)	Dens. matrice (%)	Temps fact. (s)	Temps résol. (s)
1	0,1803e+03	0,3179e−05	0,5587e+00	0,4374e+01	0,85	0,04	65,55	0,42
2	0,1803e+03	0,2402e+01	0,1171e−01	0,4374e+01	0,85	0,04	65,55	0,43
3	0,1803e+03	0,1169e+00	0,3382e−03	0,4374e+01	0,85	0,04	65,52	0,42
4	0,1803e+03	0,3855e−02	0,1014e−04	0,4374e+01	0,85	0,04	65,56	0,43
5	0,1803e+03	0,1115e−03	0,3405e−06	0,4374e+01	0,85	0,04	65,55	0,42

Méthode d'Uzawa - Algorithme LU								
Itér. Newton	$\| \vec{U} \|_2$	$\| \delta\vec{U} \|_2$	Résidu équations	Résidu masse	Dens. facteurs (%)	Dens. matrice (%)	Temps fact. (s)	Temps résol. (s)
1	0,1800e+03	0,1031e−05	0,5538e+00	0,6796e−14	0,75	0,03	44,07	0,33
2	0,1800e+03	0,2180e+01	0,8604e−02	0,2323e−09	0,75	0,03	44,06	0,34
3	0,1800e+03	0,4425e−01	0,6471e−02	0,2251e−09	0,75	0,03	44,03	0,34
4	0,1800e+03	0,1206e−02	0,2067e−03	0,1069e−10	0,75	0,03	44,07	0,34
5	0,1800e+03	0,3007e−04	0,5574e−05	0,2872e−12	0,75	0,03	44,04	0,34

Formulation mixte - Algorithme PBCGSTAB									
Itér. Newton	$\| \vec{U} \|_2$	$\| \delta\vec{U} \|_2$	Résidu équations	Résidu masse	Dens. proj. (%)	Nombre itér.	Temps fact. (s)	Temps résol. (s)	Prod. mat./vec.
1	0,1800e+03	0,3310e−06	0,5586e+00	0,1708e−14	0,01	1	0,83	0,33	2
2	0,1800e+03	0,2162e+01	0,4455e−01	0,3456e−09	0,01	2	0,86	0,96	6
3	0,1800e+03	0,4864e+00	0,3054e−02	0,3707e−09	0,01	6	0,85	1,31	12
4	0,1800e+03	0,3282e−01	0,9593e−04	0,5102e−10	0,01	8	0,86	1,60	16
5	0,1800e+03	0,1043e−02	0,1562e−05	0,2822e−11	0,01	13	0,85	2,58	26
6	0,1800e+03	0,1861e−04	0,4299e−07	0,6364e−13	0,01	32	0,85	6,21	64

Tableau 3.17 – Nombre de produits matrice-vecteur de l'algorithme PBCGSTAB en fonction des itérations de Newton pour le problème de Von Karman avec une surface libre.

Pas de temps	Itér. Newton	Prod. mat./vec.
100	1	2
	3	8
	5	52
299	1	2
	3	10
	6	68
540	1	2
	3	14
	6	66

CHAPITRE 4

EXEMPLES D'APPLICATIONS

Comme nous avons pu le constater dans le chapitre 1, la discrétisation des équations de Navier-Stokes et de transport par la méthode des éléments finis aboutit à une structure matricielle augmentée de la forme

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}. \tag{4.1}$$

À la lumière des résultats numériques présentés au chapitre 3, nous pouvons affirmer que l'utilisation des algorithmes projetés de la section 2.2.3 se révèle adéquate pour résoudre de tels systèmes linéaires. Il serait toutefois intéressant de voir si d'autres problèmes classiques d'écoulements aboutissent à des structures matricielles similaires. Ce chapitre traite des écoulements non isothermes, des fluides visco-élastiques et de la méthode des domaines fictifs. La modélisation de chacun de ces problèmes rajoute des équations supplémentaires aux équations de Navier-Stokes. Nous allons donc voir ici si les systèmes résultant préservent la structure (4.1).

4.1 Écoulements non isothermes

Lorsqu'un écoulement ne peut être considéré comme isotherme, le bilan des énergies doit être pris en compte. Ce dernier exprime, dans un volume infinitésimal $\mathcal{V}$, la loi suivante :

$$\begin{pmatrix} \textit{flux d'énergie} \\ \textit{aux parois} \\ \textit{de } \mathcal{V} \end{pmatrix} + \begin{pmatrix} \textit{flux de} \\ \textit{chaleur} \\ \textit{rejeté} \\ \textit{de } \mathcal{V} \end{pmatrix} + \begin{pmatrix} \textit{flux de la production} \\ \textit{de travail par les} \\ \textit{contraintes} \\ \textit{dans } \mathcal{V} \end{pmatrix} = \begin{pmatrix} \textit{variation} \\ \textit{d'énergie} \\ \textit{dans } \mathcal{V} \end{pmatrix}.$$

Cette loi physique se traduit mathématiquement par (Chorin and Marsden 1992) :

$$\rho c_p(\boldsymbol{u} \cdot \boldsymbol{\nabla} T) = \boldsymbol{\nabla} \cdot (k \boldsymbol{\nabla} T) + g, \tag{4.2}$$

où T est la température du fluide, c_p est la chaleur massique, k est la conductivité thermique, ρ est la densité du fluide et g est une source de chaleur.

4.1.1 Problème fort

En rajoutant l'équation (4.2) aux équations de Stokes en régime permanent, établies au chapitre 1, nous obtenons :

$$\begin{cases} -\boldsymbol{\nabla} \cdot \big(2\mu \dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big) + \boldsymbol{\nabla} p &= \rho \boldsymbol{f}\,; \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} &= 0\,; \\ \rho c_p(\boldsymbol{u} \cdot \boldsymbol{\nabla} T) - \boldsymbol{\nabla} \cdot (k \boldsymbol{\nabla} T) &= g, \end{cases} \tag{4.3}$$

auxquelles il faut ajouter les conditions aux limites

$$\begin{cases} \boldsymbol{u} &= \boldsymbol{u}_{D_{\boldsymbol{u}}} \quad \text{sur} \quad \Gamma_{D_{\boldsymbol{u}}}\,; \\ \boldsymbol{\sigma} \cdot \boldsymbol{n} &= \boldsymbol{t}_{N_{\boldsymbol{u}}} \quad \text{sur} \quad \Gamma_{N_{\boldsymbol{u}}}\,; \\ T &= T_{D_T} \quad \text{sur} \quad \Gamma_{D_T}\,; \\ -k \boldsymbol{\nabla} T \cdot \boldsymbol{n} &= q_{N_T} \quad \text{sur} \quad \Gamma_{N_T}, \end{cases} \tag{4.4}$$

et les conditions initiales

$$\begin{cases} \boldsymbol{u}(\boldsymbol{x}, t = 0) &= \boldsymbol{u}_0\,; \\ T(\boldsymbol{x}, t = 0) &= T_0. \end{cases}$$

Les domaines $\Gamma_{D_{\boldsymbol{u}}}$ et $\Gamma_{N_{\boldsymbol{u}}}$ sont respectivement les domaines d'imposition des conditions de Dirichlet et de Neumann et forment une partition de la frontière Γ. De même, nous avons :

$$\begin{cases} \Gamma_{D_T} \cup \Gamma_{N_T} &= \partial\Omega\,; \\ \Gamma_{D_T} \cap \Gamma_{N_T} &= \emptyset. \end{cases}$$

Les paramètres $\boldsymbol{u}_{D_{\boldsymbol{u}}}$, $\boldsymbol{t}_{N_{\boldsymbol{u}}}$, T_{D_T} et q_{N_T} sont donc connus.

4.1.2 Formulation variationnelle

Les formes faibles des équations de conservation de la quantité de mouvement et de conservation de la masse ont déjà été calculées au chapitre 1. Occupons-nous donc de la formulation variationnelle de l'équation de l'énergie. Considérons $g \in L^2(\Omega)$. Nous avons alors pour tout $\vartheta \in H^1_{D_T}(\Omega)$:

$$\rho c_p\big(\boldsymbol{u}\cdot\boldsymbol{\nabla}T,\vartheta\big)_{0,\Omega} - \big(\boldsymbol{\nabla}\cdot(k\boldsymbol{\nabla}T),\vartheta\big)_{0,\Omega} = \big(g,\vartheta\big)_{0,\Omega}$$

$$\Leftrightarrow$$

$$\rho c_p\big(\boldsymbol{u}\cdot\boldsymbol{\nabla}T,\vartheta\big)_{0,\Omega} + k\big(\boldsymbol{\nabla}T,\boldsymbol{\nabla}\vartheta\big)_{0,\Omega} - k\langle\boldsymbol{\nabla}T\cdot\boldsymbol{n},\vartheta\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} = \big(g,\vartheta\big)_{0,\Omega}$$

$$\Leftrightarrow$$

$$\rho c_p\big(\boldsymbol{u}\cdot\boldsymbol{\nabla}T,\vartheta\big)_{0,\Omega} + k\big(\boldsymbol{\nabla}T,\boldsymbol{\nabla}\vartheta\big)_{0,\Omega} = -\langle q_{N_T},\vartheta\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} + \big(g,\vartheta\big)_{0,\Omega}$$

On obtient finalement la forme faible des équations (4.3)-(4.4) :

$$\left\{\begin{array}{lcl} a(\boldsymbol{u},\boldsymbol{w}) - b(p,\boldsymbol{w}) & = & \langle \boldsymbol{t}_{N_u},\boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} + \rho\big(\boldsymbol{f},\boldsymbol{w}\big)_{0,\Omega}\,; \\ b(q,\boldsymbol{u}) & = & 0\,; \\ c(\boldsymbol{u},T,\vartheta) + d(T,\vartheta) & = & -\langle q_{N_T},\vartheta\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} + \big(g,\vartheta\big)_{0,\Omega} \end{array}\right. \tag{4.5}$$

où

$$\left\{\begin{array}{lcl} a(\boldsymbol{u},\boldsymbol{w}) & = & 2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}\,; \\ b(p,\boldsymbol{w}) & = & \big(p,\boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega}\,; \\ c(\boldsymbol{u},T,\vartheta) & = & \rho c_p\big(\boldsymbol{u}\cdot\boldsymbol{\nabla}T,\vartheta\big)_{0,\Omega}\,; \\ d(T,\vartheta) & = & k\big(\boldsymbol{\nabla}T,\boldsymbol{\nabla}\vartheta\big)_{0,\Omega}. \end{array}\right.$$

4.1.3 Discrétisation

La discrétisation des équations (4.5) se fait de façon similaire à celle effectuée à la section 1.2.3. Comme à la section 1.2.4, une étape de linéarisation est nécessaire. Définissons donc les trois fonctionnelles suivantes :

$$\left\{\begin{array}{lcl} R_1(\boldsymbol{u},p,\boldsymbol{w}) & = & a(\boldsymbol{u},\boldsymbol{w}) - b(p,\boldsymbol{w}) - \langle \boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} - \rho\big(\boldsymbol{f},\boldsymbol{w}\big)_{0,\Omega}\,; \\ R_2(\boldsymbol{u},q) & = & b(q,\boldsymbol{u})\,; \\ R_3(\boldsymbol{u},T,\vartheta) & = & \rho c_p\big(\boldsymbol{u}\cdot\boldsymbol{\nabla}T,\vartheta\big)_{0,\Omega} + k\big(\boldsymbol{\nabla}T,\boldsymbol{\nabla}\vartheta\big)_{0,\Omega} + \\ & & \langle q_{N_T},\vartheta\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} - \big(g,\vartheta\big)_{0,\Omega}. \end{array}\right.$$

Sachant que $(\boldsymbol{u}^n,p^n,T^n)$ est une solution approchée à l'itération n, il s'agit alors de trouver les corrections $\delta\boldsymbol{u}$, δp et δT telles que :

$$\left\{\begin{array}{lcl} R_1(\boldsymbol{u}^n+\delta\boldsymbol{u},p^n+\delta p,\boldsymbol{w}) & = & 0\,; \\ R_2(\boldsymbol{u}^n+\delta\boldsymbol{u},q) & = & 0\,; \\ R_3(\boldsymbol{u}^n+\delta\boldsymbol{u},T^n+\delta T,\vartheta) & = & 0. \end{array}\right.$$

L'utilisation du développement de Taylor autour du point $(\boldsymbol{u}^n,p^n,T^n)$ nous donne :

$$\left\{\begin{array}{lcl} 0 & = & R_1(\boldsymbol{u}^n,p^n,\boldsymbol{w}) + \left.\dfrac{\partial R_1}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u} + \left.\dfrac{\partial R_1}{\partial p}\right|_n \delta p\,; \\ 0 & = & R_2(\boldsymbol{u}^n,q) + \left.\dfrac{\partial R_2}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u}\,; \\ 0 & = & R_3(\boldsymbol{u}^n,T^n,\vartheta) + \left.\dfrac{\partial R_3}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u} + \left.\dfrac{\partial R_3}{\partial T}\right|_n \delta T. \end{array}\right.$$

En utilisant la dérivée d'une fonctionnelle telle que définie par la définition (1.38), nous avons :

$$\left\{\begin{array}{lcl} \left.\dfrac{\partial R_3}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u} & = & \rho c_p\big(\delta\boldsymbol{u}\cdot\boldsymbol{\nabla}T^n,\vartheta\big)_{0,\Omega}\,; \\ \left.\dfrac{\partial R_3}{\partial T}\right|_n \delta T & = & \rho c_p\big(\boldsymbol{u}^n\cdot\boldsymbol{\nabla}\delta T,\vartheta\big)_{0,\Omega} + k\big(\boldsymbol{\nabla}\delta T,\boldsymbol{\nabla}\vartheta\big)_{0,\Omega}. \end{array}\right.$$

Les autres dérivées ayant déjà été calculées à la section 1.2.4, on obtient ainsi la forme faible des équations linéarisées :

$$\begin{cases} a(\delta\boldsymbol{u},\boldsymbol{w}) - b(\delta p,\boldsymbol{w}) & = & -R_1(\boldsymbol{u}^n,p^n,\boldsymbol{w})\,; \\ \\ b(q,\delta\boldsymbol{u}) & = & -R_2(\boldsymbol{u}^n,q)\,; \\ \\ c(\delta\boldsymbol{u},T^n,\vartheta)+c(\boldsymbol{u}^n,\delta T,\vartheta)+d(\delta T,\vartheta) & = & -R_3(\boldsymbol{u}^n,\vartheta). \end{cases}$$

En réordonnant les équations de la façon suivante :

$$\begin{cases} a(\delta\boldsymbol{u},\boldsymbol{w}) - b(\delta p,\boldsymbol{w}) & = & -R_1(\boldsymbol{u}^n,p^n,\boldsymbol{w})\,; \\ \\ c(\delta\boldsymbol{u},T^n,\vartheta)+c(\boldsymbol{u}^n,\delta T,\vartheta)+d(\delta T,\vartheta) & = & -R_3(\boldsymbol{u}^n,T^n\vartheta)\,; \\ \\ b(q,\delta\boldsymbol{u}) & = & -R_2(\boldsymbol{u}^n,q), \end{cases} \tag{4.6}$$

on constate que la discrétisation éléments-finis des équations (4.6) aboutit à un système linéaire dont la matrice a la structure suivante :

$$\begin{pmatrix} A & 0 & B^T \\ T_1 & T_2 & 0 \\ B & 0 & 0 \end{pmatrix} = \begin{pmatrix} \mathcal{A} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{pmatrix},$$

où on a noté $\mathcal{A} = \begin{pmatrix} A & 0 \\ T_1 & T_2 \end{pmatrix}$ et $\mathcal{B} = \begin{pmatrix} B \\ 0 \end{pmatrix}$. Les matrices T_1 et T_2 résultent respectivement de la discrétisation éléments-finis des termes $c(\delta\boldsymbol{u},T^n,\vartheta)$ et $c(\boldsymbol{u}^n,\delta T,\vartheta)+d(\delta T,\vartheta)$. Nous obtenons donc une structure matricielle qui se prête bien aux algorithmes projetés présentés à la section 2.2.3.

4.2 Fluides visco-élastiques

La visco-élasticité, comme son nom l'indique, est une généralisation des théories de l'élasticité et de la viscosité. Un matériau élastique idéal réagit à l'application

d'une force par une déformation proportionnelle à cette force. Un tel matériau est représenté selon les conventions habituelles par un ressort. Un matériau visqueux (ou Newtonien) idéal réagit à l'application d'une force par un flux de vitesse constante proportionnelle à cette force et est représenté selon les conventions habituelles par un amortisseur. Vu d'une façon différente, un matériau élastique possède la capacité de stocker toute l'énergie mécanique de déformation sans dissipation, tandis qu'un fluide visqueux dissipe toute l'énergie mécanique de déformation, sans capacité d'en stocker. Un matériau visco-élastique combine ces deux propriétés. Il réagit à l'application d'une force par une déformation instantanée et un flux. Si une seconde force est appliquée, son effet se superpose à celui de la première. Ce point implique une propriété importante des matériaux visco-élastiques, qui est de conserver la mémoire des états passés : l'état d'un matériau élastique n'est déterminé que par l'état des contraintes à l'instant présent, tandis que l'état d'un matériau visco-élastique est déterminé en plus par l'ensemble des états passés des contraintes qu'il a subies.

4.2.1 Problème fort

À la base des fluides visco-élastiques, nous avons les équations de Stokes présentées au chapitre 1 :

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} - \boldsymbol{\nabla} \cdot (\boldsymbol{\tau} - p\boldsymbol{I}) &= \rho \boldsymbol{f} \, ; \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} &= 0. \end{cases} \tag{4.7}$$

Pour prendre en compte la caractéristique visco-élastique des fluides, il faut rajouter une *loi de comportement*, notée $\mathcal{H}$, qui met en relation le tenseur $\boldsymbol{\tau}$ et la vitesse $\boldsymbol{u}$, de sorte que les équations de modélisation s'écrivent :

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} - \boldsymbol{\nabla} \cdot (\boldsymbol{\tau} - p\boldsymbol{I}) &= \rho \boldsymbol{f} \, ; \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} &= 0 \, ; \\ \mathcal{H}(\boldsymbol{u}, \boldsymbol{\tau}) &= 0. \end{cases} \tag{4.8}$$

Les conditions aux limites sont :

$$\begin{cases} \boldsymbol{u} & = & \boldsymbol{u}_{D_u} & \text{sur} & \Gamma_{D_u} ; \\ \boldsymbol{\sigma} \cdot \boldsymbol{n} & = & \boldsymbol{t}_{N_u} & \text{sur} & \Gamma_{N_u}, \end{cases} \tag{4.9}$$

et la condition initiale

$$\boldsymbol{u}(\boldsymbol{x}, t = 0) \quad = \quad \boldsymbol{u}_0.$$

Les domaines Γ_{D_u} et Γ_{N_u} sont respectivement les domaines d'imposition des conditions de Dirichlet et de Neumann et forment une partition de la frontière Γ. Les paramètres $\boldsymbol{u}_{D_u}$ et $\boldsymbol{t}_{N_u}$ sont donc connus.

Plusieurs choix sont proposés pour modéliser la loi de comportement. Parmis eux, celui d'Oldroyd (Carreau et al. 1997) qui décompose le tenseur $\boldsymbol{\tau}$ comme suit :

$$\boldsymbol{\tau} = \boldsymbol{\tau}_\nu + \boldsymbol{\tau}_s,$$

où $\boldsymbol{\tau}_s = 2\mu_s \dot{\boldsymbol{\gamma}}(\boldsymbol{u})$ est le tenseur associé au solvent et $\boldsymbol{\tau}_\nu$ est la contribution viscoélastique. Ce dernier vérifie :

$$\boldsymbol{\tau}_\nu + \lambda\Big(\frac{\partial \boldsymbol{\tau}_\nu}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{\tau}_\nu - \boldsymbol{\nabla}\boldsymbol{u} \cdot \boldsymbol{\tau}_\nu - \boldsymbol{\tau}_\nu^T \cdot (\boldsymbol{\nabla}\boldsymbol{u})^T\Big) = 2\mu_\nu \dot{\boldsymbol{\gamma}}(\boldsymbol{u}). \tag{4.10}$$

où λ est le temps de relaxation et μ_ν est la viscosité de la composante élastique du fluide. De même, nous écrivons $\mu = \mu_s + \mu_\nu$ où μ_s est la viscosité du solvent. L'équation de conservation de la quantité de mouvement devient alors :

$$\frac{\partial \boldsymbol{u}}{\partial t} - \boldsymbol{\nabla} \cdot \big(\boldsymbol{\tau} - p\boldsymbol{I}\big) = \rho \boldsymbol{f}$$

$$\Leftrightarrow$$

$$\frac{\partial \boldsymbol{u}}{\partial t} - \boldsymbol{\nabla} \cdot \big(\boldsymbol{\tau}_s + \boldsymbol{\tau}_\nu - p\boldsymbol{I}\big) = \rho \boldsymbol{f}$$

$$\Leftrightarrow$$

$$\frac{\partial \boldsymbol{u}}{\partial t} - \boldsymbol{\nabla} \cdot \big(2\mu_s \dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big) + \boldsymbol{\nabla} p - \boldsymbol{\nabla} \cdot \boldsymbol{\tau}_\nu = \rho \boldsymbol{f}.$$

Ainsi, le système (4.8) devient :

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} - \boldsymbol{\nabla}\cdot\big(2\mu_s\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big) + \boldsymbol{\nabla}p - \boldsymbol{\nabla}\cdot\boldsymbol{\tau}_\nu & = & \rho\boldsymbol{f}\,; \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} & = & 0\,; \\ \boldsymbol{\tau}_\nu + \lambda\big(\dfrac{\partial \boldsymbol{\tau}_\nu}{\partial t} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{\tau}_\nu - \boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{\tau}_\nu - \boldsymbol{\tau}_\nu^T\cdot(\boldsymbol{\nabla}\boldsymbol{u})^T\big) - 2\mu_\nu\dot{\boldsymbol{\gamma}}(\boldsymbol{u}) & = & 0, \end{cases} \tag{4.11}$$

L'équation de comportement (4.10) étant hyperbolique, des conditions aux limites doivent être appliquées en amont de l'écoulement :

$$\Gamma^-(\Omega) = \{\boldsymbol{x}\in\partial\Omega \mid \boldsymbol{u}\cdot\boldsymbol{n} < 0\}, \tag{4.12}$$

où $\boldsymbol{n}$ est la normale extérieure au domaine Ω. Nous avons donc la condition

$$\boldsymbol{\tau}_\nu = \boldsymbol{\tau}_{\nu\Gamma^-} \quad \text{sur} \quad \Gamma^-(\Omega). \tag{4.13}$$

4.2.2 Formulation variationnelle

Par soucis de simplicité, et cela ne changeant pas fondamentalement la structure matricielle que nous allons obtenir, nous ne considérerons pas les dérivées partielles en temps dans le système d'équations (4.11). Considérons d'abord l'équation de conservation de la quantité de mouvement. Reprenons la formulation variationnelle (1.22) en régime permanent et en omettant le terme de convection. Pour les fluides viscoélastiques, il faut rajouter le terme $\big(\boldsymbol{\nabla}\cdot\boldsymbol{\tau}_\nu, \boldsymbol{w}\big)_{0,\Omega}$ qui devient, après intégration par parties :

$$\big(\boldsymbol{\nabla}\cdot\boldsymbol{\tau}_\nu, \boldsymbol{w}\big)_{0,\Omega} = \langle\boldsymbol{\tau}_\nu\cdot\boldsymbol{n}, \boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} - \big(\boldsymbol{\tau}_\nu, \boldsymbol{\nabla}\boldsymbol{w}\big)_{0,\Omega},$$

ce qui nous donne :

$$2\mu_s\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega} - \big(p, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega} + \big(\boldsymbol{\tau}_\nu, \boldsymbol{\nabla}\boldsymbol{w}\big)_{0,\Omega} = \langle\boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} + \rho\big(\boldsymbol{f}, \boldsymbol{w}\big)_{0,\Omega}.$$

La forme faible de l'équation de conservation de la masse est donnée par l'équation (1.26). Pour ce qui est de la loi de comportement, on utilise la méthode de Galerkin discontinue qui consiste à appliquer la méthode des éléments finis sur chaque

élément plutôt que sur tout le maillage. Pour chaque élément K, notons respectivement $\Gamma^+(K)$ et $\Gamma^-(K)$ les frontières entrantes et sortantes telles que :

$$\begin{aligned}\Gamma^+(K) &= \{\boldsymbol{x} \in \partial K \mid \boldsymbol{u}(\boldsymbol{x}) \cdot \boldsymbol{n}_K(\boldsymbol{x}) > 0\}\,;\\ \Gamma^-(K) &= \{\boldsymbol{x} \in \partial K \mid \boldsymbol{u}(\boldsymbol{x}) \cdot \boldsymbol{n}_K(\boldsymbol{x}) < 0\},\end{aligned}$$

où $\boldsymbol{n}_K$ est la normale extérieure de la frontière de l'élément K (cf. figure 4.1).

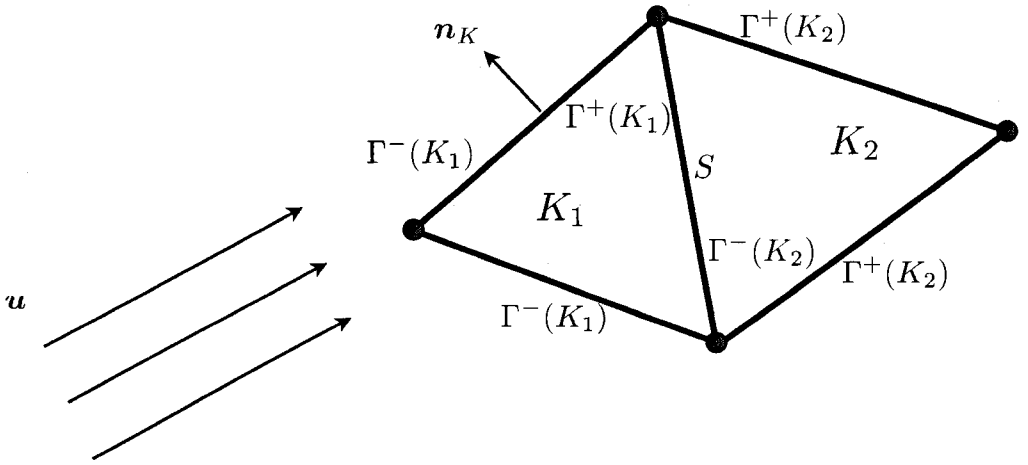


Figure 4.1 – La méthode de Galerkin discontinue.

Le saut de $\boldsymbol{\tau}_\nu$ à la frontière S de deux éléments est noté :

$$[\boldsymbol{\tau}_\nu]_S = \boldsymbol{\tau}_\nu|_{\Gamma^-(K_2)} - \boldsymbol{\tau}_\nu|_{\Gamma^+(K_1)}$$

Ainsi, pour tout tenseur de fonctions test $\boldsymbol{\varrho} \in \{\boldsymbol{\sigma} \in L^2(\Omega)^{2\times 2} \mid \boldsymbol{\sigma}_{ij} = \boldsymbol{\sigma}_{ji},\ i,j = 1,2\}$, nous avons :

$$\int_{K_2} (\boldsymbol{u} \cdot \boldsymbol{\nabla}\boldsymbol{\tau}_\nu)\boldsymbol{\tau}_\nu : \boldsymbol{\varrho}\, d\boldsymbol{x} = \int_{K_2} (\boldsymbol{u} \cdot \boldsymbol{\nabla}\boldsymbol{\tau}_\nu)\boldsymbol{\tau}_\nu : \boldsymbol{\varrho}\, d\boldsymbol{x} + \int_{\Gamma^-(K_2)} (\boldsymbol{u} \cdot \boldsymbol{n})[\boldsymbol{\tau}_\nu]_S : \boldsymbol{\varrho}\, ds.$$

Sur chaque élément K du maillage, nous aurons donc :

$$\begin{aligned}\int_K \left(\boldsymbol{\tau}_\nu + \lambda(\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{\tau}_\nu - \lambda\boldsymbol{\nabla}\boldsymbol{u} \cdot \boldsymbol{\tau}_\nu - \lambda\boldsymbol{\tau}_\nu \cdot (\boldsymbol{\nabla}\boldsymbol{u})^T - 2\mu_\nu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\right) : \boldsymbol{\varrho}\, d\boldsymbol{x} + \\ \int_{\Gamma^-(K)} (\boldsymbol{u} \cdot \boldsymbol{n})[\boldsymbol{\tau}_\nu]_S : \boldsymbol{\varrho}\, ds = 0\end{aligned} \tag{4.14}$$

En notant Ω_h le maillage éléments-finis, on obtient finalement la forme faible des équations (4.11)-(4.9) :

$$\begin{cases} a(\boldsymbol{u},\boldsymbol{w}) - b(p,\boldsymbol{w}) + \big(\boldsymbol{\tau}_\nu, \boldsymbol{\nabla w}\big)_{0,\Omega} &= \langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} + \rho\big(\boldsymbol{f},\boldsymbol{w}\big)_{0,\Omega}\,; \\ b(q,\boldsymbol{u}) &= 0\,; \\ h_K(\boldsymbol{u},\boldsymbol{\tau}_\nu,\boldsymbol{\varrho}) &= 0, \quad \forall\, K \in \Omega_h, \end{cases} \tag{4.15}$$

où

$$\begin{cases} a(\boldsymbol{u},\boldsymbol{w}) &= 2\mu_s\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}\,; \\ b(p,\boldsymbol{w}) &= \big(p, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega}\,; \\ h_K(\boldsymbol{u},\boldsymbol{\tau}_\nu,\boldsymbol{\varrho}) &= \big(\boldsymbol{\tau}_\nu + \lambda(\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{\tau}_\nu - \lambda\boldsymbol{\nabla u}\cdot\boldsymbol{\tau}_\nu - \lambda\boldsymbol{\tau}_\nu\cdot(\boldsymbol{\nabla u})^T - 2\mu_\nu\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\boldsymbol{\varrho}\big)_{0,K} \\ &\quad + \int_{\Gamma^-(K)} (\boldsymbol{u}\cdot\boldsymbol{n})[\boldsymbol{\tau}_\nu]_S : \boldsymbol{\varrho}\, ds = 0. \end{cases}$$

4.2.3 Discrétisation

La discrétisation des équations (4.15) se fait de façon similaire à celle effectuée à la section 1.2.3. Comme à la section 1.2.4, une étape de linéarisation est nécessaire. Définissons donc les trois fonctionnelles suivantes :

$$\begin{cases} R_1(\boldsymbol{u},p,\boldsymbol{\tau}_\nu,\boldsymbol{w}) &= a(\boldsymbol{u},\boldsymbol{w}) - b(p,\boldsymbol{w}) + \big(\boldsymbol{\tau}_\nu,\boldsymbol{\nabla w}\big)_{0,\Omega} \\ &\quad -\langle \boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}} - \rho\big(\boldsymbol{f},\boldsymbol{w}\big)_{0,\Omega}\,; \\ R_2(\boldsymbol{u},q) &= b(q,\boldsymbol{u})\,; \\ R_3^K(\boldsymbol{u},\boldsymbol{\tau}_\nu,\boldsymbol{\varrho}) &= h_K(\boldsymbol{u},\boldsymbol{\tau}_\nu,\boldsymbol{\varrho}), \quad \forall\, K \in \Omega_h. \end{cases}$$

Sachant que $(\boldsymbol{u}^n, p^n, \boldsymbol{\tau}_\nu^n)$ est une solution approchée à l'itération n, il s'agit alors de trouver les corrections $\delta\boldsymbol{u}$, δp et $\delta\boldsymbol{\tau}_\nu$ telles que :

$$\begin{cases} R_1(\boldsymbol{u}^n+\delta\boldsymbol{u}, p^n+\delta p, \boldsymbol{\tau}_\nu^n+\delta\boldsymbol{\tau}_\nu, \boldsymbol{w}) &= 0\,; \\ R_2(\boldsymbol{u}^n+\delta\boldsymbol{u}, q) &= 0\,; \\ R_3^K(\boldsymbol{u}^n+\delta\boldsymbol{u}, \boldsymbol{\tau}_\nu^n+\delta\boldsymbol{\tau}_\nu, \boldsymbol{\varrho}) &= 0, \quad \forall\, K \in \Omega_h, \end{cases}$$

L'utilisation du développement de Taylor autour du point $(\boldsymbol{u}^n, p^n, \boldsymbol{\tau}_\nu^n)$ nous donne :

$$\left\{\begin{array}{rcl} 0 &=& R_1(\boldsymbol{u}^n, p^n, \boldsymbol{\tau}_\nu^n, \boldsymbol{w}) + \left.\dfrac{\partial R_1}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u} + \left.\dfrac{\partial R_1}{\partial p}\right|_n \delta p + \left.\dfrac{\partial R_1}{\partial \boldsymbol{\tau}_\nu}\right|_n \delta\boldsymbol{\tau}_\nu\,; \\ 0 &=& R_2(\boldsymbol{u}^n, q) + \left.\dfrac{\partial R_2}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u}\,; \\ 0 &=& R_3^K(\boldsymbol{u}^n, \boldsymbol{\tau}_\nu^n, \boldsymbol{\varrho}) + \left.\dfrac{\partial R_3^K}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u} + \left.\dfrac{\partial R_3^K}{\partial \boldsymbol{\tau}_\nu}\right|_n \delta\boldsymbol{\tau}_\nu, \quad \forall\, K \in \Omega_h, \end{array}\right.$$

En utilisant la dérivée d'une fonctionnelle telle que définie par la définition (1.38), nous avons :

$$\left\{\begin{array}{rcll} \left.\dfrac{\partial R_1}{\partial \boldsymbol{\tau}_\nu}\right|_n \delta\boldsymbol{\tau}_\nu &=& \left(\delta\boldsymbol{\tau}_\nu, \boldsymbol{\nabla}\boldsymbol{w}\right)_{0,\Omega}\,; & \\ \left.\dfrac{\partial R_3^K}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u} &=& h_K(\delta\boldsymbol{u}, \boldsymbol{\tau}_\nu^n, \boldsymbol{\varrho}), & \forall\, K \in \Omega_h\,; \\ \left.\dfrac{\partial R_3^K}{\partial \boldsymbol{\tau}_\nu}\right|_n \delta\boldsymbol{\tau}_\nu &=& h_K(\boldsymbol{u}^n, \delta\boldsymbol{\tau}_\nu, \boldsymbol{\varrho}), & \forall\, K \in \Omega_h. \end{array}\right.$$

Les autres dérivées ayant déjà été calculées à la section 1.2.4, on obtient ainsi la forme faible des équations linéarisées :

$$\left\{\begin{array}{lcl} a(\delta\boldsymbol{u}, \boldsymbol{w}) - b(\delta p, \boldsymbol{w}) + \left(\delta\boldsymbol{\tau}_\nu, \boldsymbol{\nabla}\boldsymbol{w}\right)_{0,\Omega} &=& -R_1(\boldsymbol{u}^n, p^n, \boldsymbol{\tau}_\nu^n, \boldsymbol{w})\,; \\ b(q, \delta\boldsymbol{u}) &=& -R_2(\boldsymbol{u}^n, q)\,; \\ h_K(\delta\boldsymbol{u}, \boldsymbol{\tau}_\nu^n, \boldsymbol{\varrho}) + h_K(\boldsymbol{u}^n, \delta\boldsymbol{\tau}_\nu, \boldsymbol{\varrho}) &=& -R_3^K(\boldsymbol{u}^n, \boldsymbol{\tau}_\nu^n, \boldsymbol{\varrho}), \end{array}\right. \tag{4.16}$$

pour tout élément K du maillage. En réordonnant les différents termes de la façon suivante :

$$\left\{\begin{array}{lcl} a(\delta\boldsymbol{u}, \boldsymbol{w}) + \left(\delta\boldsymbol{\tau}_\nu, \boldsymbol{\nabla}\boldsymbol{w}\right)_{0,\Omega} - b(\delta p, \boldsymbol{w}) &=& -R_1(\boldsymbol{u}^n, p^n, \boldsymbol{\tau}_\nu^n, \boldsymbol{w})\,; \\ h_K(\delta\boldsymbol{u}, \boldsymbol{\tau}_\nu^n, \boldsymbol{\varrho}) + h_K(\boldsymbol{u}^n, \delta\boldsymbol{\tau}_\nu, \boldsymbol{\varrho}) &=& -R_3(\boldsymbol{u}^n, \boldsymbol{\tau}_\nu^n, \boldsymbol{\varrho})\,; \\ b(q, \delta\boldsymbol{u}) &=& -R_2(\boldsymbol{u}^n, q), \end{array}\right. \tag{4.17}$$

et en effectuant l'assemblage éléments-finis des formulations variationnelles obtenues sur chacun des éléments de la loi de comportement, on constate que la discrétisation éléments-finis des équations (4.17) aboutit à un système linéaire dont la matrice a la structure suivante :

$$\begin{pmatrix} A & T_\nu & B^T \\ H_1 & H_2 & 0 \\ B & 0 & 0 \end{pmatrix} = \begin{pmatrix} \mathcal{A} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{pmatrix},$$

où on a noté $\mathcal{A} = \begin{pmatrix} A & T_\nu \\ H_1 & H_2 \end{pmatrix}$ et $\mathcal{B} = \begin{pmatrix} B \\ 0 \end{pmatrix}$. Les matrices H_1 et H_2 résultent respectivement de la discrétisation éléments-finis des termes $h_K(\delta\boldsymbol{u}, \boldsymbol{\tau}_\nu^n, \boldsymbol{\varrho})$ et $h_K(\boldsymbol{u}^n, \delta\boldsymbol{\tau}_\nu, \boldsymbol{\varrho})$, et la matrice T_ν résulte de la discrétisation éléments-finis du terme $\left(\delta\boldsymbol{\tau}_\nu, \boldsymbol{\nabla w}\right)_{0,\Omega}$. Nous obtenons donc une structure matricielle qui se prête bien aux algorithmes projetés présentés à la section 2.2.3.

4.3 Méthode des domaines fictifs

La méthode des domaines fictifs est souvent utilisée pour résoudre des problèmes dans un domaine de calcul dont la géométrie évolue dans le temps. La frontière de la géométrie est discrétisée à l'aide de points de contrôle et introduite dans un domaine qui est alors plus simple à mailler. Il suffit ensuite d'imposer des conditions sur l'écoulement aux points de contrôle (Saul'ev 1963, Glowinski et al. 1994, Glowinski et al. 1999). Nous nous intéressons ici aux équations modélisant un écoulement dans un domaine comportant un obstacle mobile. Nous présentons la condition supplémentaire qui intervient dans le problème fort (1.16), la formulation variationnelle qui en découle, ainsi que le nouveau système linéaire issu de la discrétisation éléments-finis.

4.3.1 Problème fort

Soit $\Omega^* \subset \Omega$ un obstacle de frontière Γ^* tel qu'illustré à la figure 4.2.

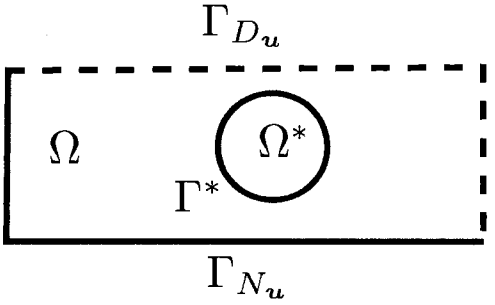


Figure 4.2 – Introduction d'un obstacle Ω^* dans un domaine de calcul Ω.

En régime permanent, les équations de Stokes modélisant l'écoulement sont alors :

$$\begin{cases} -\boldsymbol{\nabla} \cdot \left(2\mu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\right) + \boldsymbol{\nabla} p &= \rho \boldsymbol{f} \quad \text{dans } \Omega \backslash \Omega^* \, ; \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} &= 0 \quad \text{dans } \Omega \backslash \Omega^*, \end{cases} \tag{4.18}$$

auxquelles il faut ajouter les conditions aux limites

$$\begin{cases} \boldsymbol{u} &= \boldsymbol{u}_{D_u} \quad \text{sur} \quad \Gamma_{D_u} \, ; \\ \boldsymbol{\sigma} \cdot \boldsymbol{n} &= \boldsymbol{t}_{N_u} \quad \text{sur} \quad \Gamma_{N_u} \, ; \\ \boldsymbol{u} &= \boldsymbol{u}^* \quad \text{sur} \quad \Gamma^*, \end{cases} \tag{4.19}$$

et la condition initiale

$$\boldsymbol{u}(\boldsymbol{x}, t = 0) = \boldsymbol{u}_0.$$

En plus des conditions aux limites (1.5), on ajoute une condition de Dirichlet supplémentaire sur la vitesse au bord de l'obstacle Ω^*. Les domaines Γ_{D_u} et Γ_{N_u} sont respectivement les domaines d'imposition des conditions de Dirichlet et de Neumann et forment une partition de la frontière Γ. Les paramètres $\boldsymbol{u}^*$, $\boldsymbol{u}_{D_u}$ et $\boldsymbol{t}_{N_u}$ sont donc connus.

4.3.2 Formulation variationnelle

Lorsque Ω^* évolue avec le temps, il n'est pas concevable, en terme de coût de calcul, de reconstruire un nouveau maillage à chaque pas de temps. C'est pour contourner cette difficulté que l'on utilise la méthode des *domaines fictifs*. Le principe de la

méthode repose sur la construction de deux maillage $\mathcal{T}_1$ et $\mathcal{T}_2$:

- le maillage $\mathcal{T}_1$ est une discrétisation du domaine Ω. L'obstacle Ω^* est ignoré. Si ce dernier se déplace, $\mathcal{T}_1$ restera fixe. Ce maillage n'est donc construit qu'une seule fois ;
- le maillage $\mathcal{T}_2$ est une discrétisation de la frontière Γ^*. On utilise pour cela des *points de contrôle*. Ces derniers peuvent être vus, en $2D$, comme un maillage linéique (cf. figure 4.3).

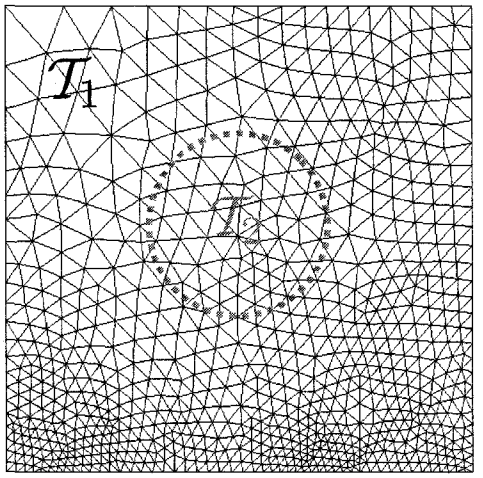


Figure 4.3 – Discrétisation de la frontière Γ^* de l'obstacle à l'aide de points de contrôle.

Il est donc important de noter que l'on construit deux maillages indépendants, le premier pour le domaine Ω et le deuxième pour la frontière Γ^*.

Afin d'intégrer la nouvelle contrainte dans la formulation variationnelle du système d'équations (4.18) - (4.19), nous allons voir ce dernier comme un problème d'optimisation avec contraintes :

$$\begin{cases} -\boldsymbol{\nabla} \cdot (2\mu\dot{\boldsymbol{\gamma}}(\boldsymbol{u})) + \boldsymbol{\nabla} p &= \rho \boldsymbol{f} \quad \text{dans } \Omega\,; \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} &= 0 \quad \text{dans } \Omega. \end{cases} \tag{4.20}$$

Les équations (4.20) peuvent être vues comme les conditions d'optimalité au premier ordre du problème de minimisation (Fortin and Glowinksy 1983) :

$$\begin{cases} \min\limits_{\boldsymbol{u}} & \mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big)_{0,\Omega} - \rho\big(\boldsymbol{f},\boldsymbol{u}\big)_{0,\Omega} - \langle \boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{u}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}\ ; \\ \text{s.c.} & \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0. \end{cases} \tag{4.21}$$

Cette réécriture du problème (4.20) est possible car la forme bilinéaire $\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big)_{0,\Omega}$ est symétrique définie positive. Notons $L(\boldsymbol{u},p)$ le lagrangien correspondant au problème (4.21) :

$$L(\boldsymbol{u},p) = \mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big)_{0,\Omega} - \big(p,\boldsymbol{\nabla}\cdot\boldsymbol{u}\big)_{0,\Omega} - \rho\big(\boldsymbol{f},\boldsymbol{u}\big)_{0,\Omega} - \langle \boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{u}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}},$$

où p est le multiplicateur de Lagrange associé à la contrainte $\boldsymbol{\nabla}\cdot\boldsymbol{u} = 0$. Ajoutons à présent l'obstacle Ω^* et donc la contrainte sur la vitesse imposée sur la frontière Γ^*. On a donc à résoudre le problème :

$$\begin{cases} \min\limits_{\boldsymbol{u}} & \mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big)_{0,\Omega} - \rho\big(\boldsymbol{f},\boldsymbol{u}\big)_{0,\Omega} - \langle \boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{u}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}\ ; \\ \text{s.c.} & \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0\ ; \\ & \boldsymbol{u}|_{\Gamma^*} = \boldsymbol{u}^*. \end{cases} \tag{4.22}$$

Le lagrangien correspondant est donné par :

$$L^*(\boldsymbol{u},p,\boldsymbol{\lambda}) = L(\boldsymbol{u},p) + \big(\boldsymbol{\lambda},\boldsymbol{u}-\boldsymbol{u}^*\big)_{\Gamma^*},$$

où $\boldsymbol{\lambda}$ est le vecteur dont les composantes sont les multiplicateurs de Lagrange associés à la deuxième contrainte du problème (4.22). Les conditions d'optimalité du premier ordre du problème faible (4.22) sont alors :

$$\begin{cases} \dfrac{d}{d\alpha}L^*(\boldsymbol{u}+\alpha\boldsymbol{w},p,\boldsymbol{\lambda})\Big|_{\alpha=0} &= 0, \quad \forall\, \boldsymbol{w}\ ; \\ \dfrac{d}{d\alpha}L^*(\boldsymbol{u},p+\alpha q,\boldsymbol{\lambda})\Big|_{\alpha=0} &= 0, \quad \forall\, \boldsymbol{q}\ ; \\ \dfrac{d}{d\alpha}L^*(\boldsymbol{u},p,\boldsymbol{\lambda}+\alpha\boldsymbol{\varsigma})\Big|_{\alpha=0} &= 0, \quad \forall\, \boldsymbol{\varsigma}. \end{cases}$$

Ainsi, la première dérivation nous donne

$$\frac{d}{d\alpha}L^*(\boldsymbol{u}+\alpha\boldsymbol{w},p,\boldsymbol{\lambda})\Big|_{\alpha=0}=0$$

$$\Leftrightarrow$$

$$\frac{d}{d\alpha}\big[\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}+\alpha\boldsymbol{w}),\dot{\boldsymbol{\gamma}}(\boldsymbol{u}+\alpha\boldsymbol{w})\big)_{0,\Omega}-\big(p,\boldsymbol{\nabla}\cdot(\boldsymbol{u}+\alpha\boldsymbol{w})\big)_{0,\Omega}-\rho\big(\boldsymbol{f},\boldsymbol{u}+\alpha\boldsymbol{w}\big)_{0,\Omega}$$
$$-\langle\boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{u}+\alpha\boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}+\big(\boldsymbol{\lambda},\boldsymbol{u}+\alpha\boldsymbol{w}-\boldsymbol{u}^*\big)_{\Gamma^*}\big]_{\alpha=0}=0$$

$$\Leftrightarrow$$

$$\frac{d}{d\alpha}\big[\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big)_{0,\Omega}+\alpha^2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{w}),\dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}+2\alpha\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}$$
$$-\big(p,\boldsymbol{\nabla}\cdot\boldsymbol{u}\big)_{0,\Omega}-\alpha\big(p,\boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega}-\rho\big(\boldsymbol{f},\boldsymbol{u}\big)_{0,\Omega}-\alpha\rho\big(\boldsymbol{f},\boldsymbol{w}\big)_{0,\Omega}$$
$$-\langle\boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{u}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}-\alpha\langle\boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}+\big(\boldsymbol{\lambda},\boldsymbol{u}-\boldsymbol{u}^*\big)_{\Gamma^*}+\alpha\big(\boldsymbol{\lambda},\boldsymbol{w}\big)_{\Gamma^*}\big]_{\alpha=0}=0$$

$$\Leftrightarrow$$

$$2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}-\big(p,\boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega}-\rho\big(\boldsymbol{f},\boldsymbol{w}\big)_{0,\Omega}-\langle\boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{w}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}+\big(\boldsymbol{\lambda},\boldsymbol{w}\big)_{\Gamma^*}=0,$$

la seconde,

$$\frac{d}{d\alpha}L^*(\boldsymbol{u},p+\alpha q,\boldsymbol{\lambda})\Big|_{\alpha=0}$$

$$\Leftrightarrow$$

$$\frac{d}{d\alpha}\big[\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big)_{0,\Omega}-\big(p+\alpha q,\boldsymbol{\nabla}\cdot\boldsymbol{u}\big)_{0,\Omega}-\rho\big(\boldsymbol{f},\boldsymbol{u}\big)_{0,\Omega}-\langle\boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{u}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}$$
$$+\big(\boldsymbol{\lambda},\boldsymbol{u}-\boldsymbol{u}^*\big)_{\Gamma^*}\big]_{\alpha=0}=0$$

$$\Leftrightarrow$$

$$\frac{d}{d\alpha}\big[\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}),\dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big)_{0,\Omega}-\big(p,\boldsymbol{\nabla}\cdot\boldsymbol{u}\big)_{0,\Omega}-\alpha\big(q,\boldsymbol{\nabla}\cdot\boldsymbol{u}\big)_{0,\Omega}-\rho\big(\boldsymbol{f},\boldsymbol{u}\big)_{0,\Omega}-$$
$$\langle\boldsymbol{t}_{N_{\boldsymbol{u}}},\boldsymbol{u}\rangle_{H^{-\frac{1}{2}},H^{\frac{1}{2}}}+\big(\boldsymbol{\lambda},\boldsymbol{u}-\boldsymbol{u}^*\big)_{\Gamma^*}\big]_{\alpha=0}=0$$

$$\Leftrightarrow$$

$$\big(q,\boldsymbol{\nabla}\cdot\boldsymbol{u}\big)_{0,\Omega}=0,$$

et la troisième,

$$\frac{d}{d\alpha} L^*(\boldsymbol{u}, p, \boldsymbol{\lambda} + \alpha\boldsymbol{\varsigma})\Big|_{\alpha=0} = 0$$

$$\Leftrightarrow$$

$$\frac{d}{d\alpha}\big[\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big)_{0,\Omega} - \big(p, \boldsymbol{\nabla}\cdot\boldsymbol{u}\big)_{0,\Omega} - \rho\big(\boldsymbol{f}, \boldsymbol{u}\big)_{0,\Omega} - \langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{u}\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}}$$
$$+\big(\boldsymbol{\lambda} + \alpha\boldsymbol{\varsigma}, \boldsymbol{u} - \boldsymbol{u}^*\big)_{\Gamma^*}\big]_{\alpha=0} = 0$$

$$\Leftrightarrow$$

$$\frac{d}{d\alpha}\big[\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{u})\big)_{0,\Omega} - \big(p, \boldsymbol{\nabla}\cdot\boldsymbol{u}\big)_{0,\Omega} - \rho\big(\boldsymbol{f}, \boldsymbol{u}\big)_{0,\Omega} - \langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{u}\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}}$$
$$+\big(\boldsymbol{\lambda}, \boldsymbol{u} - \boldsymbol{u}^*\big)_{\Gamma^*} + \alpha\big(\boldsymbol{\varsigma}, \boldsymbol{u} - \boldsymbol{u}^*\big)_{\Gamma^*}\big]_{\alpha=0} = 0$$

$$\Leftrightarrow$$

$$\big(\boldsymbol{\varsigma}, \boldsymbol{u} - \boldsymbol{u}^*\big)_{\Gamma^*} = 0.$$

On obtient finalement :

$$\begin{cases} a(\boldsymbol{u}, \boldsymbol{w}) - b(p, \boldsymbol{w}) + \big(\boldsymbol{\lambda}, \boldsymbol{w}\big)_{\Gamma^*} &= \langle \boldsymbol{t}_{N_{\boldsymbol{u}}}, \boldsymbol{w}\rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} + \rho\big(\boldsymbol{f}, \boldsymbol{w}\big)_{0,\Omega}\,; \\ b(q, \boldsymbol{u}) &= 0\,; \\ \big(\boldsymbol{\varsigma}, \boldsymbol{u}\big)_{\Gamma^*} &= \big(\boldsymbol{\varsigma}, \boldsymbol{u}^*\big)_{\Gamma^*}, \end{cases} \tag{4.23}$$

où

$$\begin{cases} a(\boldsymbol{u}, \boldsymbol{w}) &= 2\mu\big(\dot{\boldsymbol{\gamma}}(\boldsymbol{u}), \dot{\boldsymbol{\gamma}}(\boldsymbol{w})\big)_{0,\Omega}\,; \\ b(p, \boldsymbol{w}) &= \big(p, \boldsymbol{\nabla}\cdot\boldsymbol{w}\big)_{0,\Omega}. \end{cases}$$

Comparativement au problème présenté à la section 1.2.2, on note l'ajout du terme supplémentaire $\big(\boldsymbol{\lambda}, \boldsymbol{w}\big)_{\Gamma^*}$ dans la formulation variationnelle de l'équation de mouvement, ainsi que de l'équation $\big(\boldsymbol{u}, \boldsymbol{\varsigma}\big)_{\Gamma^*} = \big(\boldsymbol{u}^*, \boldsymbol{\varsigma}\big)_{\Gamma^*}$.

4.3.3 Discrétisation

La discrétisation des équations (4.23) se fait de façon similaire à celle effectuée à la section 1.2.3. Seuls trois termes supplémentaires sont à prendre en compte :

$$\left(\boldsymbol{\lambda}, \boldsymbol{w}\right)_{\Gamma^*}, \quad \left(\boldsymbol{u}, \boldsymbol{\varsigma}\right)_{\Gamma^*} \quad \text{et} \quad \left(\boldsymbol{u}^*, \boldsymbol{\varsigma}\right)_{\Gamma^*}.$$

Comme nous l'avons expliqué plus tôt, la frontière Γ^* est discrétisée à l'aide de points de contrôle (cf. figure 4.3). Considérons ces derniers au nombre de N^c. Comme à la section 1.2.4, une étape de linéarisation est nécessaire. Définissons donc les trois fonctionnelles suivantes :

$$\begin{cases} R_1(\boldsymbol{u}, p, \boldsymbol{\lambda}, \boldsymbol{w}) &= a(\boldsymbol{u}, \boldsymbol{w}) - b(p, \boldsymbol{w}) + \left(\boldsymbol{\lambda}, \boldsymbol{w}\right)_{\Gamma^*} \\ & \quad -\langle \boldsymbol{t}_{N_u}, \boldsymbol{w} \rangle_{H^{-\frac{1}{2}}, H^{\frac{1}{2}}} - \rho\left(\boldsymbol{f}, \boldsymbol{w}\right)_{0,\Omega} ; \\ R_2(\boldsymbol{u}, q) &= b(q, \boldsymbol{u}) ; \\ R_3(\boldsymbol{u}, \boldsymbol{\varsigma}) &= \left(\boldsymbol{u}, \boldsymbol{\varsigma}\right)_{\Gamma^*} - \left(\boldsymbol{u}^*, \boldsymbol{\varsigma}\right)_{\Gamma^*}. \end{cases}$$

Sachant que $(\boldsymbol{u}^n, p^n, \boldsymbol{\lambda}^n)$ est une solution approchée à l'itération n, il s'agit alors de trouver les corrections $\delta\boldsymbol{u}$, δp et $\delta\boldsymbol{\lambda}$ telles que :

$$\begin{cases} R_1(\boldsymbol{u}^n + \delta\boldsymbol{u}, p^n + \delta p, \boldsymbol{\lambda}^n + \delta\boldsymbol{\lambda}, \boldsymbol{w}) &= 0 ; \\ R_2(\boldsymbol{u}^n + \delta\boldsymbol{u}, q) &= 0 ; \\ R_3(\boldsymbol{u}^n + \delta\boldsymbol{u}, \boldsymbol{\varsigma}) &= 0. \end{cases}$$

L'utilisation du développement de Taylor autour du point $(\boldsymbol{u}^n, p^n, \boldsymbol{\lambda}^n)$ nous donne :

$$\begin{cases} 0 &= R_1(\boldsymbol{u}^n, p^n, \boldsymbol{\lambda}^n, \boldsymbol{w}) + \left.\dfrac{\partial R_1}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u} + \left.\dfrac{\partial R_1}{\partial p}\right|_n \delta p + \left.\dfrac{\partial R_1}{\partial \boldsymbol{\lambda}}\right|_n \delta\boldsymbol{\lambda} ; \\ 0 &= R_2(\boldsymbol{u}^n, q) + \left.\dfrac{\partial R_2}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u} ; \\ 0 &= R_3(\boldsymbol{u}^n, \boldsymbol{\varsigma}) + \left.\dfrac{\partial R_3}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u}. \end{cases}$$

En utilisant la dérivée d'une fonctionnelle telle que définie par la définition (1.38), nous avons :

$$\begin{cases} \left.\dfrac{\partial R_1}{\partial \boldsymbol{\lambda}}\right|_n \delta\boldsymbol{\lambda} &= \left(\delta\boldsymbol{\lambda}, \boldsymbol{w}\right)_{\Gamma^*} ; \\ \\ \left.\dfrac{\partial R_3}{\partial \boldsymbol{u}}\right|_n \delta\boldsymbol{u} &= \left(\delta\boldsymbol{u}, \boldsymbol{\varsigma}\right)_{\Gamma^*}, \end{cases}$$

Les autres dérivées ayant déjà été calculées à la section 1.2.4, on obtient ainsi la forme faible des équations linéarisées :

$$\begin{cases} a(\delta\boldsymbol{u}, \boldsymbol{w}) - b(\delta p, \boldsymbol{w}) + \left(\delta\boldsymbol{\lambda}, \boldsymbol{w}\right)_{\Gamma^*} &= -R_1(\boldsymbol{u}^n, p^n, \boldsymbol{\lambda}^n, \boldsymbol{w}) ; \\ \\ b(q, \delta\boldsymbol{u}) &= -R_2(\boldsymbol{u}^n, q) ; \\ \\ \left(\boldsymbol{\varsigma}, \delta\boldsymbol{u}\right)_{\Gamma^*} &= -R_3(\boldsymbol{u}^n, \boldsymbol{\varsigma}). \end{cases} \tag{4.24}$$

En effectuant la discrétisation éléments-finis de la formulation variationnelle (4.24), on obtient en régime permanent un système linéaire dont la matrice a la structure suivante :

$$\begin{pmatrix} A & B^T & D^T \\ B & 0 & 0 \\ D & 0 & 0 \end{pmatrix} = \begin{pmatrix} A & \mathcal{B}^T \\ \mathcal{B} & 0 \end{pmatrix}, \tag{4.25}$$

où on a noté $\mathcal{B} = \begin{pmatrix} B \\ D \end{pmatrix}$. La matrice D résulte de la discrétisation éléments-finis du terme $(\boldsymbol{\varsigma}, \delta\boldsymbol{u})_{\Gamma^*}$. Nous obtenons donc une structure matricielle qui se prête bien aux algorithmes projetés présentés à la section 2.2.3.

Remarque : pour traiter les problèmes transitoires, avec un obstacle en mouvement, il faut inclure au système matriciel (4.25) les termes transitoires et utiliser un schéma de discrétisation en temps approprié comme expliqué à la section 1.2.4.1.

Nous n'effectuerons pas cette étape afin de ne pas alourdir inutilement les équations.

Nous constatons donc dans ce chapitre que plusieurs problèmes classiques d'écoulements de fluides aboutissent à des systèmes linéaires dont la matrice possède la structure (4.1). L'utilisation des méthodes itératives projetées se révèle par conséquent adéquate. Il est également possible d'envisager des écoulements plus complexes en « combinant » les différents problèmes rencontrés dans ce chapitre. Nous obtenons donc un écoulement non isotherme, visco-élastique et rencontrant un obstacle modélisé par la méthode des domaines fictifs. La matrice finale obtenue aura alors la structure suivante :

$$\begin{pmatrix} A & 0 & T_\nu & B^T & D^T \\ T_1 & T_2 & 0 & 0 & 0 \\ H_1 & 0 & H_2 & 0 & 0 \\ B & 0 & 0 & 0 & 0 \\ D & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \mathcal{A} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{pmatrix},$$

où on a noté $\mathcal{A} = \begin{pmatrix} A & 0 & T_\nu \\ T_1 & T_2 & 0 \\ H_1 & 0 & H_2 \end{pmatrix}$ et $\mathcal{B} = \begin{pmatrix} B & 0 & 0 \\ D & 0 & 0 \end{pmatrix}$. Cette matrice possède encore la structure (4.1).

CONCLUSION

Pour conclure ce mémoire, nous pouvons affirmer que l'objectif principal que nous nous sommes fixé est atteint, c'est-à-dire l'implémentation et les tests d'une nouvelle classe de méthodes itératives permettant de résoudre efficacement les équations de Stokes et de Navier-Stokes. Plus exactement, ces méthodes sont à mi-chemin entre les méthodes directes et itératives étant donné qu'elles nécessitent la factorisation de la matrice de projection (cf. équation (2.24)). Dans un premier temps, nous avons considéré les équations de Stokes en régime stationnaire. Ces dernières correspondent à des écoulements à faible nombre de Reynolds. La discrétisation par la méthode des éléments finis mène à un système linéaire qui peut être résolu par l'algorithme du gradient conjugué projeté. En effet, la matrice obtenue (cf. équation (1.43)) a une structure augmentée dont le bloc A est symétrique défini positif. Il est donc possible d'effectuer une analogie avec les problèmes d'optimisation quadratique avec contraintes d'égalité (Glowinski and Le Tallec 1987). Les résultats alors obtenus sur les problèmes de Poiseuille et du jet immergé à un fluide ont pu être comparés à ceux fournis par les méthodes directes. Nous avons ainsi pu constater les gains importants réalisés en termes de temps de calcul et de stockage mémoire par rapport aux méthodes directes.

Dans un second temps, nous nous sommes penchés sur les écoulements qui, après discrétisation des équations par la méthode des éléments finis, aboutissent à des systèmes linéaires dont la matrice n'est plus symétrique. Il n'y a alors plus d'interprétation en terme de problème d'optimisation quadratique avec contraintes d'égalité et l'algorithme du gradient conjugué projeté ne peut dès lors plus être utilisé. Il nous a donc fallu utiliser un algorithme adapté à la structure augmentée des matrices. Nous avons pour cela sélectionné, parmis les algorithmes classiques fréquemment utilisés, le plus performant. Nous avons donc effectué une série de tests et nous avons choisi l'algorithme BiCGSTAB qui s'est avéré être le plus rapide et le moins gour-

mand en stockage mémoire. Nous en avons ensuite testé la version projetée, PBCG-STAB (Orban 2008). Appliquée au problème des allées de Von Karmann, nous avons réalisé encore une fois d'importants gains en rapidité et en mémoire.

Enfin, dans un dernier temps, nous avons établi trois problèmes d'écoulement pouvant être résolus grâce aux méthodes itératives projetées :

- les écoulements non isothermes, comme leur nom l'indique, nécessitent la discrétisation de l'équation de conservation de l'énergie. La température vient donc se rajouter à la vitesse et à la pression comme inconnue du problème ;
- pour ce qui est des fluides visco-élastiques, une loi de comportement, qui lie le tenseur des contraintes au champ de vitesse, vient s'ajouter aux équations de Navier-Stokes. Cette équation supplémentaire traduit le fait que la déformation de ces fluides n'est plus directement proportionnelle à la force appliquée ;
- La méthode des domaines fictifs permet de modéliser un obstacle dans le domaine d'écoulement à l'aide de points de contrôle. Ces derniers sont indépendants du maillage éléments-finis. D'un point de vue de la modélisation mathématique, cela se traduit par des contraintes supplémentaires imposées sur la vitesse aux points de contrôle.

Pour chacun de ces trois problèmes, la discrétisation par la méthode des éléments finis entraîne l'apparition de blocs matriciels supplémentaires. Cependant, ces nouveaux blocs n'altèrent pas la structure augmentée des matrices et les algorithmes projetés peuvent donc toujours être utilisés.

L'originalité de ce projet de recherche réside dans l'implémentation et la vérification d'une nouvelle classe de méthodes itératives particulièrement adaptée aux matrices issues de la discrétisation éléments-finis des équations de Navier-Stokes. Contrairement aux méthodes usuelles, les méthodes projetées tirent avantage du bloc matriciel de 0 apparaissant lors de la discrétisation de l'équation de conservation de la masse.

De nombreuses extensions de ce mémoire sont possibles :

- mentionnons tout d'abord la mise en place d'une bibliothèque d'algorithmes projetés. En effet, nous avons implémenté la version projetée de l'algorithme BiCGSTAB et des travaux sont également en cours sur l'algorithme TFQMR (Orban 2008). Il est donc envisageable de généraliser ce procédé aux autres algorithmes populaires tels GMRES ou MINRES ;
- ensuite, nous pourrions implémenter les trois applications présentées au chapitre 4 afin de mesurer la performance des méthodes itératives projetées. En effet, les matrices des problèmes que nous avons considérées au chapitre 3 ont un conditionnement de l'ordre de 10^6, ce qui reste relativement raisonnable. L'ajout de l'équation de l'énergie (cf. équations (4.2)) pour les fluides non isothermes, ou de la loi de comportement (cf. équations (4.10)) pour les fluides visco-élastiques, aurait sans doute pour conséquence d'augmenter le conditionnement et donc de rendre la convergence des algorithmes plus « difficile » ;
- une autre extension possible est la programmation d'une routine permettant d'effectuer les produits matrice-vecteur avec A (cf. équation (1.43)) sans avoir à la former explicitement. En effet, les algorithmes projetés n'ont besoin explictement que de la matrice B pour former la matrice de projection et la factoriser. Notons au passage qu'une fois la factorisation effectuée, le stockage mémoire de la matrice B n'est plus nécessaire ;
- l'étude de préconditionneurs adéquats fait également partie des éventuels travaux futurs. Cela prendrait son importance lors de la résolution de problèmes plus complexes, tels les écoulements turbulents qui aboutissent à des matrices au conditionnement très élevé ;
- enfin, une autre mise en oeuvre intéressante est l'utilisation des méthodes projetées aux problèmes 3D qui requièrent un stockage mémoire important.

RÉFÉRENCES

Agassant, J. F., Fortin, A., et Demay, Y. (1994). Prediction of stationary interfaces in coextrusion flows. *Polym. Eng. Sci.*, **34**(14), 1101–1108.

Arrow, K., Hurwicz, L., et Uzawa, H. (1958). *Studies in linear and non-linear programming.* Standford University Press.

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., et Van der Vorst, H. (1994). *Templates for the solution of linear systems : building blocks for iterative methods.* SIAM, Philadelphia, PA, second edition.

Batchelor, G. K. (1967). *An introduction to fluid dynamics.* Cambridge University Press.

Bramley, R. (1992). An orthogonal projection algorithm for linear Stokes problems. Technical Report 98-019, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign. Rapport technique.

Brooks, A. N. et Hughes, T. J. R. (1990). Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Eng.*, pages 199–259.

Carreau, P. J., De Kee, D. C. R., et Chhabra, R. P. (1997). *Rheology of polymeric systems, principles and applications.* Hanser/Gardner Publications, Cincinnati, OH.

Choquet, R. (1995). *Etude de la méthode de Newton-GMRES.* Ph.d. thesis, Université de Rennes 1, Rennes.

Chorin, A. et Marsden, J. (1992). *A mathematical introduction to fluid mechanics.* Springer-Verlag, New York, third edition.

Dutto, L. (1990). Application de l'algorithme GMRES à la résolution des équations de Navier-Stokes compressibles. Technical Report RR-1234, INRIA. Rapport technique - Unité de recherche INRIA-Rocquencourt.

Fletcher, R. (1976). *Conjugate gradient methods for indefinite systems*, pages 73–89. Lecture Notes in Mathematics. Springer Berlin/Heidelberg.

Fortin, M. et Glowinksy, R. (1983). *The augmented Lagrangian method.* Amsterdam, North-Holland edition.

Freund, R., Golub, G., et Nachtigal, N. (1992). Iterative solution of linear systems. In Iserles, A., editor, *Acta Numerica 1992*, volume 1, pages 57–100. Cambridge University Press.

Freund, R. W. et Nachtigal, N. M. (1991). QMR : a quasi-minimal residual method for non-hermitian linear systems. *Numer. Math.*, **60**(1), 151–154.

Freund, R. W. et Nachtigal, N. M. (1994). An implementation of the QMR method based on coupled two-term recurrences. *SIAM J. on Sc. Comput.*, **15**(2), 381–384.

Gibbs, N., Poole, W., et Stockmeyer, P. (1974). An algorithm for reducing the bandwith and profile of a sparse matrix. *SIAM J. Numer. Anal.*, **13**(2), 236–250.

Glowinski, R. et Le Tallec, P. (1987). *Augmented-Lagrangian and operator-splitting methods in nonlinear mechanics.* Society for Industrial and Applied Mathematics.

Glowinski, R., Pan, T. W., Hesla, T. I., et Joseph, D. D. (1999). A distributed Lagrange multiplier/fictitious domain method for particulate flows. *Int. J. of Multiphase Flow*, **25**(5), 755–794.

Glowinski, R., Pan, T. W., et Periaux, J. (1994). A fictitious domain method for Dirichlet problems and applications. *Comp. Meth. in Appl. Mech. and Eng.*, **111**, 283–303.

Gould, N. I. M., Hribar, M., et Nocedal, J. (1998). On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM J. Sci. Comput.*, **23**(4), 1376–1395.

Gresho, P., Sani, R., et Engelman, M. (1999). *Incompressible flow and the finite element method : advection-diffusion and isothermal laminar flow.* Wiley.

Harwell (2000). Harwell subroutine library : a collection of Fortran codes for large-scale scientific computation. Technical report, AERE Harwell Laboratory. www.numerical.rl.ac.uk/hsl.

Hestenes, M.R.and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *J. of research of the national bureau of standards*, **49**(6).

Johnson, C. (1990). Adaptive finite element methods for diffusion and convection problems. *Comput. Methods Appl. Mech. Eng.*, **82**(1-3), 301–322.

Lanczos, C. (1952). Solution of systems of linear equations by minimized iterations. *J. of research of the national bureau of standards*, **49**(1), 33–53.

Orban, D. (2008). Projected Krylov methods for unsymmetric augmented systems. Technical Report G-2008-46, GERAD, Montréal. cahier du GERAD.

Ralston, A. et Rabinowitz, P. (1978). *A first course in Numerical Analysis*. Dover Publications, second edition.

Saad, Y. (1990). SPARSKIT : A basic tool kit for sparse matrix computations. Technical Report 90-20, NASA AMES Research Center, Moffett Field, CA.

Saad, Y. (2000). *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, second edition.

Saad, Y. et Schultz, M. (1984). GMRES : a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, **7**(3), 856–869.

Saad, Y. et Wu, K. (1995). DQGMRES : a quasi minimal residual algorithm based on incomplete orthogonalization. Technical Report 9RR UMSI 93/131, Computer Science Department, University of Minneapolis.

Saul'ev, V. (1963). On the solution of some boundary value problem on high performance computers by fictitious domain method. *Siberian Math. Journal*, **4**(4), 912–925.

Thompson, E. (1986). Use of pseudo-concentration to follow creeping viscous flows during transient analysis. *Int. J. Numer. Methods Fluids*, **6**(10), 749–761.

Trefethen, L. et Bau, D. (1997). *Numerical Linear Algebra.* SIAM.

Van Der Vorst, H. (1992). BICGSTAB : a fast and smoothly converging variant for BiCG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, **13**(2), 631–644.